KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutus

Kim Ordén

MOBIILITASOHYPPELYPELIN IMMERSION JA KONTROLLIEN TOTEUTTAMISEN HAASTEET

Opinnäytetyö
Marraskuu 2015

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Marraskuu 2015
Tietojenkäsittelyn koulutus

Karjalankatu 3
80220 JOENSUU
013 260 600

Tekijä(t)
Kim Ordén

Nimeke
Mobiilitasohyppelypelin immersion ja kontrollien toteuttamisen haasteet

Toimeksiantaja
-

Tiivistelmä

Tässä opinnäytetyössä keskityttiin keskeisimpiin ongelmiin kontrollien ja immersion toteuttamisessa mobiilitasohyppelypeliä kehittäessä. Työssä pyrittiin myös selvittämään millaiset kontrollimenetelmät on mahdollista toteuttaa mobiilitasohyppelypelissä. Tämän tyyppisiin menetelmiin kuuluu esimerkiksi virtuaaliset painikkeet, jotka on aseteltu ruudulle simuloimaan fyysisiä painikkeita. Immersiolla viitataan siihen, miten hyvin pelaaja pystyy uppoutumaan sisälle pelimaailmaan. Immersiota pyritään luomaan peleissä yleisesti esimerkiksi grafiikoilla, tarinalla ja äänillä tai musiikilla.

Vastaukset näihin ongelmiin pyrittiin löytämään pelitestauksen ja toiminnallisen työn kautta. Pelitestauksessa kokeiltiin ja analysoitiin suosittuja mobiilitasohyppelypelejä ja toiminnallisessa osuudessa kehitettiin mobiilitasohyppelypeli. Työssä peilattiin suosittujen mobiilipelien toteutusmenetelmiä kehitettyyn tasohyppelypeliin ja perusteltiin suunnittelueroavuuksien ilmetessä valintoja sekä motiiveja.

Työssä selvisi, että mobiilitasohyppelypelissä kontrollit voidaan toteuttaa virtuaalisilla painikkeilla, ilman että kontrollien reaktioherkkyys kärsii. Selvisi myös, että mobiilitasohyppelypelin kehittäjät eivät kiinnitä paljon huomiota immersioon, vaikka se on muilla alustoilla keskeinenkin elementti peliä suunnitellessa.

Kieli
suomi

Sivuja 47
Liitteet 0
Liitesivumäärä 0

Asiasanat
mobiilipeli, tasohyppely, immersio, kontrollit

Karelia AMMATTIKORKEAKOULU

THESIS
November 2015
Degree Programme In Business Information Technology

Karjalankatu 3
80220 JOENSUU
FINLAND
013 260 600

Tekijä

Kim Ordén

Title
The Challenges of Designing controls and Creating Immersion for a Mobile Platformer game

Commissioned by -

Abstract

The aim of this thesis was to identify the central problems of implementing controls and creating immersion for a mobile platformer game. Additionally, this thesis aspires to determine the types of control schemes that are feasible on a mobile device. These types of control schemes are, for instance virtual buttons, which are situated on the screen to simulate actual physical buttons. Immersion refers to the level of the player's involvement in the game world. Immersion is exuviated into a game, for example, through a story, visuals and vast soundscapes or an enchanting score.

Solutions to these problems were strived to be found through game testing and a functional part. In the game testing part, popular mobile platformer games were tested, and in the practical part, a mobile platformer game was made. The methods of implementation were compared between the tested popular games and the mobile platformer game that was produced for this thesis.

The outcome of the research was that instantiation of virtual controls in a mobile platformer game can be executed well, without losing the reactivity of the controls. It also became clear that mobile platformer developers doesn't pay much attention on immersion in their games in general. Nevertheless, in designing games for other platforms it is an element in high value.

Language	Pages 47
Finnish	Appendices 0

Asiasanat

mobile game, platformer, immersion, controls

Sisältö

1 Johdanto

Tässä opinnäytetyössä analysoin viiden eri suositun Androidille tehdyn mobiilitasohyppelypelin kontrolleiden toteutusmenetelmiä sekä miten niissä on pyritty luomaan immersiota. Vertaan kyseisten pelien kehittäjien toteutusmenetelmiä, toiminnalliseen työhömme joka on Murgorilla-niminen tasohyppelypeli mobiilialustoille jonka työstämistä aloitimme Aki Kupiaisen kanssa vuonna 2013. Pelissä on hyödynnetty proseduurillista generaatiota kenttien luomiseen.

Keskityn kontrolleihin, koska hahmon ohjattavuus on mobiilipelissä tärkeää. Jos pelihahmo ei reagoi räväkästi pelaajan antamiin komentoihin, voi se helposti pilata koko pelikokemuksen tasohyppelypelissä, vaikka peli muilta osa-alueiltaan olisikin hyvin toteutettu. Tarkastelen myös immersiota mobiilipeleissä, koska tähän asti Immersio mobiilipeleissä on kehityksessä ilmennyt toissijaisena asiana, vaikka se on muissa peleissä tärkeä asia.

Oma motivaationi kyseisen oppinäytetyön työstämiseen lähti siitä, että tasohyppelypelit ovat olleet pitkään suosiossa. Koin myös mobiilimarkkinat mahdollisuutena. Fingersoft-niminen peliyritys oli juuri lyönyt läpi Hill Climb Racing -nimisellä pelillään ja ajattelin, että jos Fingersoft pystyi menestymään niin miksi emme me? Monet kysyvät usein, pystymmekö me menestymään, kun oikea kysymys tulisi olla, miksi emme pystyisi. Koin myös, että mobiilipeleissä kokemus saattoi jäädä konseptin puolesta joskus hieman suppeaksi ja halusin tästä syystä luoda pelikokemuksen, jonka maailmaan pystyy uppoutumaan hyvin.

Peliämme on testautettu niin ystävillä ja tuttavilla, kuin myös Scifest-tapahtumassa Joensuun Areenalla 2014 keväällä, missä yli 50 henkeä kävi kokeilemassa peliämme messutapahtumassa. Kyseisestä tapahtumasta ei ole dokumentaatiota, sillä testaus toteutettiin messujen muun esittelyn ohessa, eikä tapahtuman pääasiallinen tarkoitus ollut testauttaa peliämme, vaan lähinnä esitellä sitä potentiaalisille rahoittajille, jotka myöskin kävivät tapahtumassa tarkastelemassa peliämme.

Tämän opinnäytetyön lukemisesta on hyötyä kaikille, jotka ovat kiinnostuneita mobiilipelien kehittämisestä sekä yleisesti tasohyppelypelien kehittämisestä. Tämä opinnäytetyö on kirjoitettu aloittelevan pelinkehittäjän näkökulmasta ja teoksessa ilmi tuodut asiat auttavat todennäköisesti aloittelevia pelinkehittäjiä enemmän kuin veteraaneja. Toisaalta henkilö joka on aiemmin kehittänyt pelejä muille alustoille, mutta on kiinnostunut mobiilipelien kehityksestä, voisi saada irti kyseisestä teoksesta myös jotain. Pyrin myös kyseisellä opinnäytetyöllä herättämään mielenkiintoa yleisesti pelinkehitystä kohtaan ja tuomaan esille, miten helppoa on nykypäivänä ryhtyä pelinkehittäjäksi.

Tuon esille myös erilaisia vaihtoehtoja eri pelimoottoreille ja mitä taloudellisia tai kehitykseen liittyviä hyötyjä jokaisen kohdalla on. Rahoituksen saaminen aloittelevana pelinkehittäjänä ilman kattavaa ansiolistaa voi olla hyvin hankalaa, joten taloudellinen aspekti on todella tärkeä huomioitavaksi.

Työn tarkoituksena on tuoda esille, mitkä ovat keskeisimmät haasteet pyrkiessä kehittämään immersiivistä mobiilitasohyppelypelikokemusta ja mitä vaatimuksia käytettävälle pelimoottorille tulisi asettaa sitä valittaessa.

Opinnäytetyö alkaa kontrollien testauksella, missä käyn läpi lukuisten menestyneiden mobiilitasohyppelypelien eri kontrollintoteutusmenetelmiä. Tuon esille kyseisten pelien kehittäjien tekemiä ratkaisuja ja miten he ovat päättäneet toteuttaa kontrollit ja vertaan sitä toteutusmenetelmiimme. Analysoin myös kyseisten pelien konsepteja ja miten he ovat pyrkineet luomaan immersiota peleihinsä ja miten paljon painoarvoa he ovat kyseiselle aspektille laittaneet. Pyrin myös löytämään vastauksia siihen miten paljon konseptin laajuus tai immersio korreloi pelin menestykseen mobiilimarkkinoilla.

Seuraavaksi käyn lävitse ratkaisumenetelmämme ja miten olemme teknillisesti toteuttaneet eri mekaniikat, joita olemme päätyneet käyttämään. Perustelen eri työkalujen valintaa ja mitä muita työkaluja olisi mahdollisesti voinut käyttää. Tuon esille myös, miten päädyimme eri toteutusmenetelmiin ja miten ideat syntyivät kyseisen pelin suhteen.

Viimeisessä osiossa tuon esille pohdintaa pelimoottoritarjonnan tilanteesta nykypäivänä, sekä miten olisimme voineet toteuttaa tietyt asiat eri tavalla nykytietämyksellä. Pyrin myös tuomaan esille, mitä olemme virheittemme kautta oppineet pelinkehityksessä ja mihin mahdollisesti pitää keskittyä enemmän tulevaisuudessa, kun lähtee kehittämään uusia projekteja.

2 Murgorilla-peli ja sen idea

Pelissä on tarkoitus päästä maailmasta toiseen. Jokaisen maailman lopussa on normaalia perusvihollista voimakkaampi vihollisjohtaja joka on päihitettävä edetäkseen pelissä. Peli ei pysähdy lainkaan maailmojen välissä, mutta pelaaja voi hävitessään jatkaa siitä maailmasta, johon on parhaimmillaan päässyt. Pelaaja saa pisteitä sen mukaan, miten pitkälle hän pääsee häviämättä ja kuinka paljon vihollisia hän tappaa matkan varrella. Pelihahmon toimintoihin kuuluvat:

- Hyppy
- Sukellus (osuessaan vihollishahmoon tässä moodissa pelaaja tuhoaa vihollisen ja hyppää takaisin ilmoihin)
- Aseen heitto projektiilin muodossa, joka kimpoaa takaisin osuessaan viholliseen. Projektiili matkaa ruudun toiseen reunaan, jos osumaa ei tapahdu. Tässä tilanteessa projektiili lähtee suuntautumaan kohti pelaajaa, mutta pelaaja voi hallita sen sijaintia y-akselilla, riippuen miten pelaaja liikuttaa hahmoa.

Pelaaja kerää voimakkaammilta vihollisilta uusia varusteita, jotka vaikuttavat hahmon suorituskykyyn. Mikään varuste ei suoranaisesti paranna hahmoa, vaan muuttaa hieman hahmon käyttäytymistä ja antaa pelaajalle mahdollisuuden vaikuttaa siihen, millä tavalla haluaa hahmoaan pelata. Mitä pitemmälle pelaaja etenee pelissä, sen haastavammaksi pelin viholliset muuttuvat, mutta palkinnotkin paranevat sen mukaan.

Pelissä voi myös valita vaikeustason, jotta kaikille löytyisi mieluisa tapa pelata. Haluamme tosin palkita vaikeammilla vaikeustasoilla palaavia enemmän, joten jokaisella vaikeustasolla tippuu erilaisia tavaroita ja tietyt tavarat on mahdollista saada vain vaikeimmalla vaikeustasolla.

3 Pelien kontrollien testaus ja analysointi

Tässä osiossa käyn läpi, miten kontrollit on toteutettu viidessä suositussa mobiilitasohyppelypelissä ja analysoin niiden toteutusmenetelmiä. Suurin osa peleistä löytyvät useista ”parhaat Android-alustan tasohyppelypelit” listoilta (Hindy 2014; Timsina 2015). Otan myös huomioon toimintojen määrän ja miten sulavasti ne toimivat kokonaisuutena pelissä.

Arvioni perustan pelitestaukseen jonka toteutin kyseisen osion sivussa. Pelasin keskimäärin 30 minuuttia jokaista analysoimaani peliä saadakseni hyvän kuvan kontrollien toteutusmenetelmistä. Huomioin myös onko pelissä turvauduttu virtuaalisiin painikkeisiin, vai onko kontrollit toteutettu muulla tavalla.

3.1.1 Badland

Badland on suomalaisen Frogmind Games -nimisen yrityksen kehittämä mobiilitasoloikka(Frogmind Games 2015), jossa pelaaja ohjaa mysteeristä metsäolentoa ja hänen päämääränään on selvitä tasosta seuraavaan. Kontrollit kyseisessä pelissä ovat hyvin yksinkertaiset, sillä peliä ohjataan yhdellä painalluksella. Kun pelaaja painaa ruutua (kuva 1), hahmo nousee ylemmäksi, mutta jos sormen irrottaa ruudusta hahmo tippuu takaisin maanpinnalle. Pelin vaikeuskäyrä muodostuu siitä, että pelaaja voi jäädä jumiin esineisiin, jotka ovat levitettyinä kentälle ja tästä johtuen pelihahmo joutuu ruudun ulkopuolelle, jolloin peli päättyy. Pelaaja

kerää myös pelin edetessä enemmän näitä mysteerisiä metsäolentoja mukaansa, jotka antavat pelaajalle uusia yrityskertoja, koska kentän läpäisemiseksi riittää, että yksi hahmo pääsee maaliin saakka.

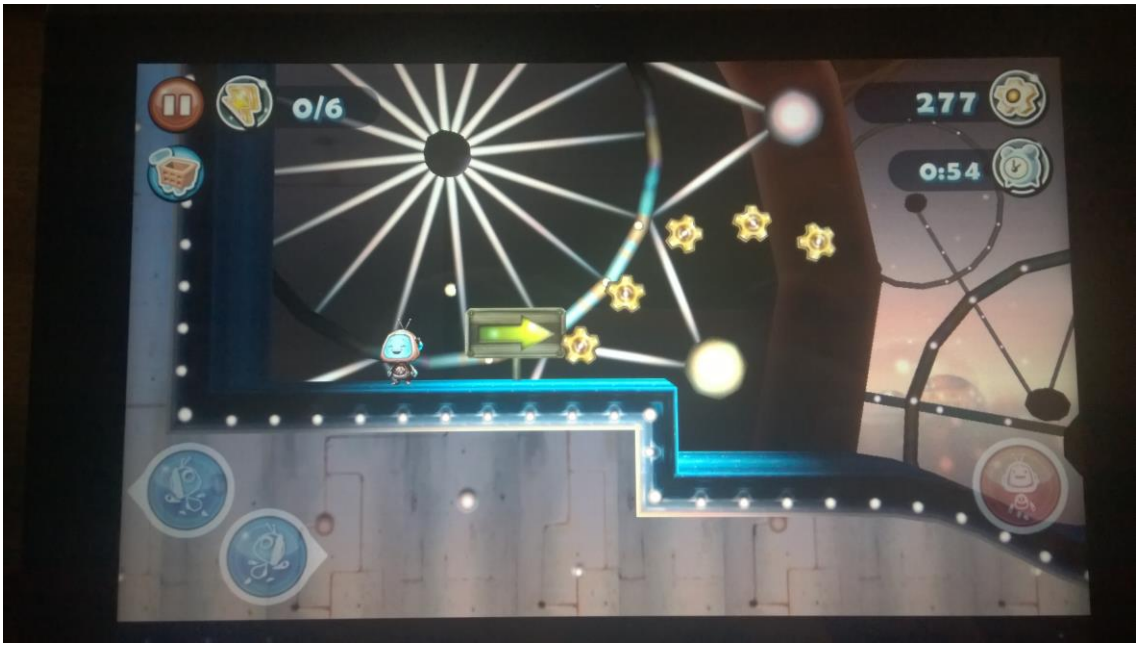
Kuva 1. Badland-pelin testausta tablettitietokoneella (Kuva: Kim Ordén).

Pelin kontrollit ovat todella sulavat yksinkertaisuutensa vuoksi ja ne on todella helppo oppia. Tästä johtuen pelissä saa helposti käsityksen mitä siinä on tehtävä. Hahmon liikkuvuutta on tosin välillä hankala arvioida(Leray 2013), koska hahmon reagoiminen pelaajan painallukseen muuttuu pelin aikana riippuen mitä päivityksiä on käytössä.

3.1.2 Cordy 2

Cordy 2 on Amerikan Kaliforniassa perustetun SilverTree Media -nimisen yrityksen kehittämä mobiilitasohyppelypeli (SilverTree Media 2015). Pelin idea on lähestulkoon sama kuin Badland-pelissä, mutta pelikenttiin on sijoiteltu kerättäviä objekteja. Pelissä on siis tarkoitus läpäistä tasoja, joissa pelaaja kohtaa vihollisia ja keräilee esineitä joista saa pisteitä. Pelin kontrollit on toteutettu hyvin eritavalla kuin Badland-pelissä, sillä pelissä käytetään nk. virtuaalikontrolleja(kuva 2), jossa

pelinäytölle yritetään simuloida fyysisen ohjaimen painikkeet. Painikkeita on pelissä kolme, joista kahdella ohjataan liikkumista oikealla tai vasemmalle ja viimeisellä hahmon saa hyppäämään. Jos painiketta painaa uudestaan ilmassa, hahmo suorittaa tuplahypyn.

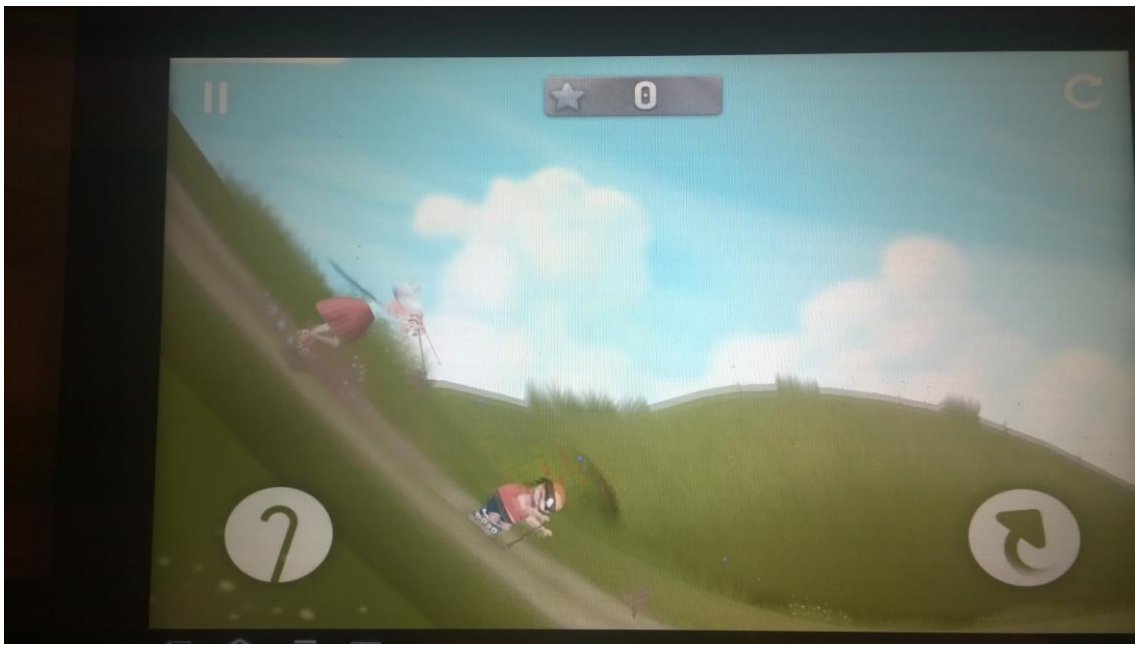


Kuva 2. Cordy2-pelin testausta tablettitietokoneella (Kuva: Kim Ordén).

Vaikka pelin kontrollit on toteutettu virtuaalipainikkeilla, ovat ne silti todella sulavan oloiset. Hahmo reagoi hyvin komentoihin, eikä tule tunnetta, että epäonnistumiset johtuvat heikosta ohjattavuudesta. Huomioitava asia on myös horisontaalia liikkumista ohjaavien painikkeiden asettelu, jossa ne eivät ole vierekkäin, vaan hieman lomittain. Tämä luo todella ergonomisen pelikokemuksen, sillä sormi liikkuu sujuvasti näppäinten välillä.

3.1.3 Granny Smith

Granny Smith on Medriocre-nimisen studion kehittämä mobiilitasohyppely-peli(Mediocre 2012), jossa pelaajan tavoitteena on päästä kentän alusta maali-viivalle saakka, samalla keräämällä tarvittavan määrän omenoita edetäkseen seuraavaan kenttään. Pelin kontrollit on toteutettu kahdella virtuaalipainik-keella(kuva 3), joilla pelaaja voi hypätä, sekä roikkua kävelykepillään kentälle asetelluissa köysissä. Pelihahmo liikkuu itsestään horisontaalissa suunnassa. Pelaajalla on ainoastaan mahdollisuus vaikuttaa siihen, milloin hän haluaa hy-pätä/roikkua köysissä. Kun pelaaja on suorittanut hypyn, pelihahmo alkaa pyöri-mään myötäpäivään, riippuen siitä miten pelaaja painaa hyppynappia pohjassa. Jos pelaaja ei ohjaa pelihahmoa laskeutumaan jaloilleen, pelihahmo kaatuu ja pelihahmolla menee hetki nousta ylös, jolloin pelaaja menettää aikaa.

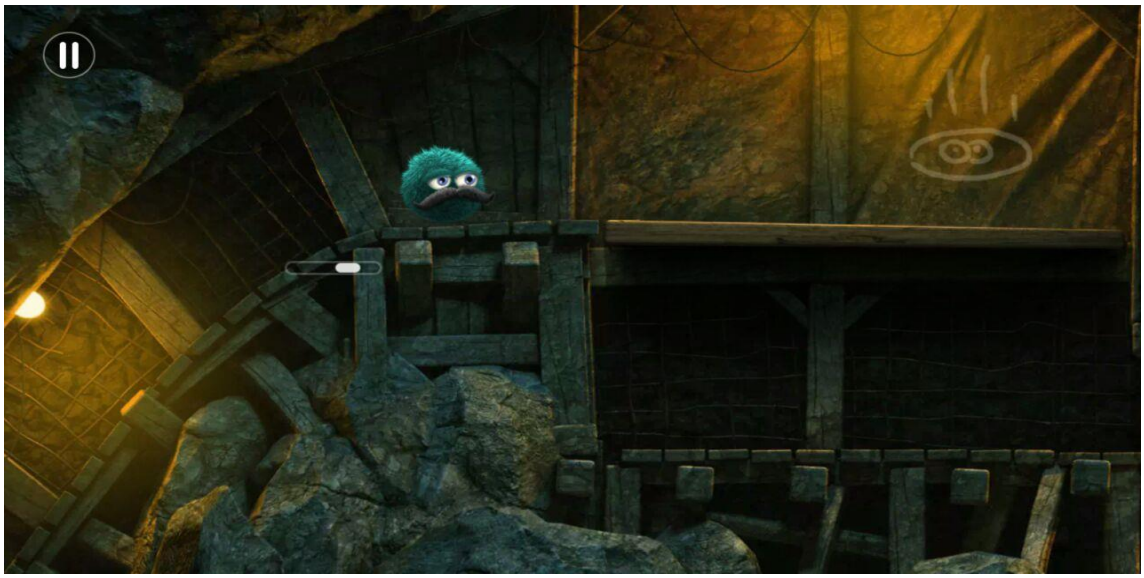


Kuva 3. Granny Smith -pelin testausta tablettitietokoneella (Kuva: Kim Ordén).

Granny Smith -pelin kontrollit ovat myös todella simppelit ja toimivat. niiden oppi-minen ei vie paljoakaan aikaa, koska toimintoja on kaksi. Kontrollit tuntuvat to-della herkiltä, sillä hahmo reagoi nopeasti painalluksiin. Pelissä ei tule tunnetta, että hävitessään virhe olisi pelissä, eikä pelaajassa.

3.1.4 Leo’s Fortune

Leo’s Fortune on ruotsalaisen 1337 & Senri LLC -nimisen yrityksen mobiilitasohyppely/seikkailupeli(1337 & Senri LLC 2014). Pelin idea on edetä tason alusta loppuun asti (kuva 4) ja kerätä matkan varrella kolikoita. Hahmon ohjattavuus rajoittuu pelissä horisontaaliseen liikkumiseen. Pelaaja voi myös laajentaa hahmoa, jolloin se hyppää tai syöksyy maata kohti. Jos pelaaja pitää hyppäämisen jälkeen sormea näytöllä, pelihahmo siirtyy liitoon, jolloin se tippuu hitaammin alaspäin. Hahmon liikuttaminen suoritetaan vetämällä sormea ruudun vasemmalla puoliskolla horisontaalissa suunnassa, kun taas hahmon laajentaminen suoritetaan ruudun oikealla puoliskolla.



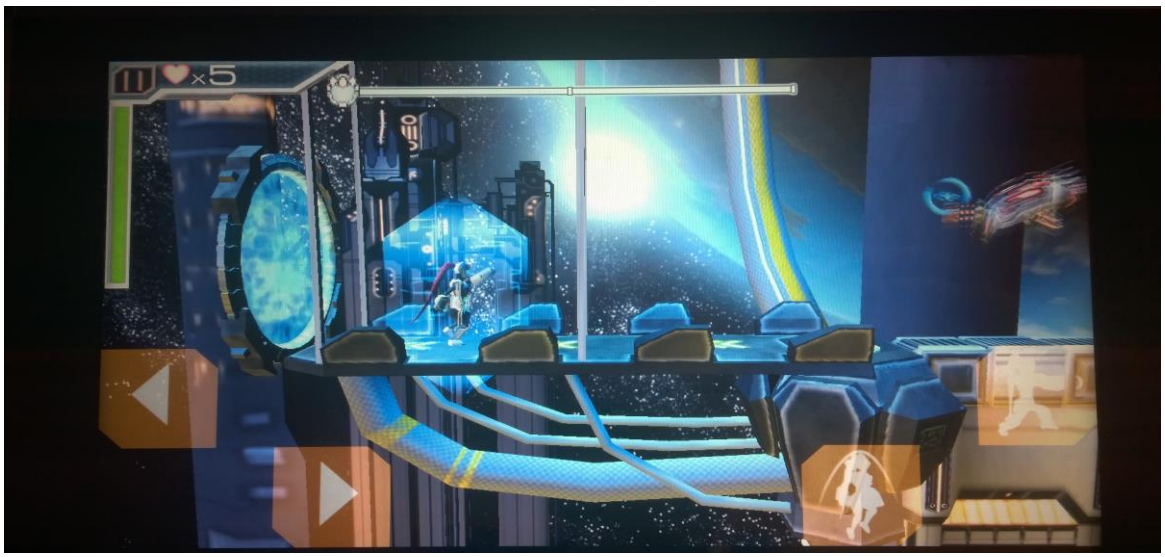
Kuva 4. Kuvankaappaus pelistä Leo’s Fortune mobiilipuhelimesta (Kuva: Kim Ordén).

Pelin ohjattavuus vaikutti aluksi hieman omituiselta, mutta tuntui viimeistellyltä. Hetken pelaamisen jälkeen kontrolleihin pääsi hyvin sisälle. Se teki mielestäni tämän pelin ohjattavuudesta paremman kuin aikaisemmista peleistä, joissa hahmoa pystyi ohjaamaan horisontaalisessa suunnassa. Syynä tähän oli se, että aikaisemmista peleistä poiketen ei ollut kiinteitä näppäimiä, vaan ruudulla näkyvä palkki päivitti positiotansa riippuen siitä, mihin pelaaja laittaa sormensa alas.

Tämä antoi pelistä todella viimeistellyn kuvan ja hahmon ohjaamistapa oli myös todella poikkeava useimpiin samankaltaisiin peleihin verrattuna.

3.1.5 Megatroid

Megatroid on Triolith Entertainment -nimisen yrityksen kehittämä mobiilitasohyppelypeli(Triolith Entertainment 2013), jossa tarkoituksena on päästä kentän alusta loppuun(kuva 6). Pelikentän aikana pelaaja kohtaa vihollisia joita pitää ampua, sekä muita esteitä. Pelissä hahmoa voi ohjata horisontaalissa suunnassa, sekä hypätä ja ampua. Ohjaaminen suoritetaan virtuaalisilla painikkeilla. Vaakasuunnassa hahmon liikuttaminen tapahtuu ruudun vasemmalla puoliskolla sijaitsevia nuolinäppäimiä painamalla(kuva 5), kun taas oikealla puolella löytyy hyppysekä ampumisnappi. Kun pelaaja on ilmassa ja hyppynappia painetaan uudelleen, hahmo suorittaa toisen hypyn.



Kuva 5. Megatroid-pelin testausta tablettitietokoneella (Kuva: Kim Ordén)

Megatroid-pelin ohjattavuus oli lähestulkoon identtinen Cordy 2 -pelin kanssa ja tuntuma oli tässäkin pelissä todella hyvä ja reagoiva. Painikkeita oli kyseisessä pelissä yksi enemmän kuin Cordy 2 -pelissä ja ne oli aseteltu ruudulle samalla tavalla kuin horisontaalia liikkumista ohjaavat nuolinäppäimet. Ampuminen ja

hyppiminen onnistuivat ongelmitta suurimmaksi osaksi pelikertaa, mutta nopeatempoisemmissa kohdissa, joissa joutui hyppimään ja ampumaan samanaikaisesti todella ripeällä vauhdilla, ohjaaminen tuntui hyvin hankalalta.

3.1.6 Mikey Shorts

Mikey Shorts on Noodlecake Studios Inc -nimisen yrityksen kehittämä mobiilitasohyppelypeli(Noodlecake Studios Inc 2015), jossa tarkoitus on päästä kentän alusta loppuun. Kenttien aikana pelaaja kerää kolikoita ja vapauttaa patsaiksi muuttuneita hahmoja kirouksestaan. Pelaaja voi ohjata hahmoa virtuaalisilla painikkeilla. Hahmon toiminta on rajattu horisontaaliseen liikkumiseen, hyppyyn sekä liukumiseen. Horisontaalin liikkeen ohjaaminen tapahtuu näytön vasemmanpuoleisista nuolipainikkeista, kun taas hyppy tapahtuu sinisestä painikkeesta ja liukuminen punaisesta. Jos hyppynappia painaa pitempään pohjassa, hahmo hyppää normaalia korkeammalle. Liukumista käytetään mm. esteiden ali liikkumiseen ja vihollisten tuhoamiseen.

Kyseisessä pelissä hahmon liikuttaminen toimi sulavasti ja suunnan vaihtaminen onnistui ripeästi ilman viiveaikoja. Peli on hyvin nopeatempoinen, mutta responsiivisten kontrolliensa ansiosta peliä oli mieluisa pelata. Pelin horisontaali liikkuminen oli toteutettu eri tavalla kuin muissa läpikäydyissä peleissä, mutta se tuntui silti luontevalta.

Kuva 6. Mikey shorts -pelin testausta tablettitietokoneella (Kuva: Kim Ordén)

3.2 Kontrollien toteutus Murgorilla-pelissä ja tarkastelluissa peleissä

Tässä osiossa vertailen edellä mainittujen pelien kontrollitoteutuksia Murgorilla-peliin. Yritän perustella, miksi meidän tekemämme ratkaisut sopivat paremmin meidän peliin, jos olemme päättäneet ratkaista joitakin kontrolleihin liittyviä ongelmia eri tavalla kun läpi käydyissä peleissä.

Olemme yrittäneet hioa käyttäjäkokemusta mahdollisimman sulavaksi, koska tiedostan kontrollien sulavuuden tärkeyden tämän tyyppisessä pelissä. Tästä syystä olen peliä suunnitellessa halunnut lähtökohtaisesti irtaantua virtuaalisista painikkeista, sillä peli on todella nopeatempoinen ja virtuaaliset painikkeet saattavat aiheuttaa haluttua enemmän virhepainalluksia. Tämä voi johtaa pelaajan turhautumiseen, mikä ei ole haluttu tila pelaajalle, sillä peli on jo itsessään haastava ja se ei saa tuntua epäreilulta kontrollien suhteen. Pelin kehityksen edetessä tulimme tosin siihen lopputulokseen, että jos haluaa toteuttaa jokseenkin tarkan ampumisjärjestelmän(pelin alkuvaiheessa hahmo pystyi ampumaan pelaajan määrittämään suuntaan), niin virtuaalisiin kontrolleihin oli turvauduttava. Lukuisten pelaajakokeilujen jälkeen tulimme siihen tulokseen, että ampumismekaniikka

ei ole kovin aloittelijaystävällinen ja mekaniikan opetteluun menee liian paljon aikaa. Kyseinen mekaniikka ei myöskään tukenut kovinkaan paljon pelin reaktiopohjaista hahmonohjausta ja tästä syystä lähdimme etsimään parempia ratkaisuja, jolloin päädyimme nykyisiin ominaisuuksiin.

Pelaaja ei voi ohjata hahmon liikkumista horisontaalissa suunnassa, mutta pystyy määrittämään milloin hahmo hyppää, sukeltaa tai heittää aseensa pitkälle etäisyydelle. Hahmo hyppää, kun pelaaja painaa ruudun vasenta puoliskoa. Kun pelaaja painaa uudelleen ruudun vasenta puoliskoa hahmon ollessa ilmassa, hahmo syöksyy takaisin tasoa kohti kovalla vauhdilla. Tällä tavalla pelaaja voi hallita todella herkästi hahmoa vertikaalisella akselilla. Kaikki vahingonteko tapahtuu ruudun oikealla puoliskolla yksinkertaisella näpäytyskomennolla. Kun pelaaja haluaa heittää aseensa pitemmälle etäisyydelle, hän näpäyttää ruudun oikeaa puoliskoa. Tästä syystä pelaaja ei koskaan tarvitse pelaamista varten enemmän kuin kaksi sormea ja kontrollien omaksumisen ei pitäisi aloittelijallekaan olla liian hankalaa.

3.3 Kontrollien vertailu

Tässä osiossa tuon esille taulukkomuodossa, mitä ominaisuuksia on käytetty missäkin peleissä. Se antaa selkeän kuvan siitä, miten useita toimintoja yhdessäkin pelissä on käytetty. Taulukossa 1 alimpana on meidän kehittämämme peli.

Taulukko 1. Pelien kontrolliominaisuudet.

Pelin nimi	Virtuaalipainikkeet	Toimintojen määrä	Vapaa ohjattavuus jokaisella akselilla	Hipaisukomennot
Badland	Ei	1	Ei	Ei
Cordy 2	Kyllä	3	Kyllä	Ei
Granny Smith	Kyllä	2	Ei	Ei
Leo’s Fortune	Ei	3	Kyllä	Kyllä
Megatroid	Kyllä	3	Kyllä	Ei
Mikey Shorts	Kyllä	3	Kyllä	Ei
Murgorilla	Ei	3	Ei	Ei

Yleisin ero tekemämme pelin ja tässä työssä tarkasteltujen pelien välillä on se, että hyppääminen hoidetaan lähestulkoon aina ruudun oikealta puolelta, kun taas meidän tekemässämme pelissä se tapahtuu ruudun vasemmalta puoliskolta. Syy tähän on se, että kun kehitimme peliä, aluksi käytimme tähtäämiseen virtuaalista

tattia, jonka kuitenkin poistimme myöhemmin pelistä. Virtuaalisen tatin ohjaaminen tuntui luontevammalta oikealla kädellä. Tätä toteutustapaa käytetään yleensä esimerkiksi konsoliräiskintä-peleissä (tähtäys toteutetaan oikeaa kättä käyttäen).

Olimme jo niin tottuneita hypyn ohjaamiseen vasemmalla kädellä, että sen vaihtaminen toiselle puolelle tuntui absurdilta ratkaisulta ja olisi vaatinut paljon aikaa kontrollien uudelleenoppimiseen, jotta testausta olisi jatkossa voinut toteuttaa tehokkaasti.

Virtuaalisilla painikkeilla toteutetut kontrollit voi myös saada toimimaan mobiilialustalla ilman, että menettää tuntumaa kokonaisuudessaan. Virtuaaliset painikkeet voivat tosin huonosti toteutettuna pilata pelikokemuksen ja tästä syystä tehdä pelistä todella epämiellyttävän pelata. Tämä tosin pätee suurimpaan osaan tasohyppelypeleistä, vaikka alusta olisikin eri. Jos kontrollit on toteutettu kehnosti, on pelistä todella hankala nauttia, vaikka se olisikin muilta osa-alueilta hyvin toteutettu. Tämä ilmeni esimerkiksi Megatroid-pelissä, josta kerron myöhemmin enemmän.

Mikey Shorts- ja Cordy 2 –peleissä virtuaalikontrollit on toteutettu hyvin. Kontrollit toimivat sulavasti ja tuntuivat pelin tempoon nähden riittävän reagoivilta. Kuten Mark Brown arvostelussaan(Brown 2012) sanoo, tuntuu siltä että, pelaajalla on täydellinen kontrolli Mikey-hahmon jokaiseen liikkeeseen. Pystyn yhtymään tähän mielipiteeseen.

Huonoiten virtuaalipainikkeilla toteutetuista kontrolleista toimi ehdottomasti Megatroid. Vaikka lähtökohtaisesti kontrollit oli toteutettu suhteellisen hyvin, ongelma ilmeni pelin nopeatempoisesta yleisluonteesta. Vaikeissa tilanteissa tuntui siltä, että virheet tapahtuivat aina siitä syystä, että ohjattavuus ei ollut tarpeeksi reagoiva. Viitaten Nissa Campbellin arvosteluun Touch Arcade sivulla (Campbell 2012). Arvostelussaan hän puhuu virtuaalisten painikkeiden kirouksesta. Tämä tarkoittaa sitä, että pelihahmo hyppää kun haluaa ampua, ja ampuu kun pelaaja haluaa hypätä. Tämä saattaa johtaa virheisiin, jotka aiheuttavat pelaajan epäonnistumisen pelissä.

Mobiilialustalle kehittäessä pelissä tapahtuvien toimintojen määrä on teknisistä syistä rajoitettava paljon pienempään määrään kuin esim. nykyaikaisessa konsolitasoloikassa, mutta jos käyttää kosketusnäytön tuomia etuja hyväksi, hahmon ohjattavuudesta voi saada yllättävänkin monipuolisen. Tämä on verrattavissa todella varhaisten konsolien peleihin, jolloin toimintapainikkeiden määrä oli rajattu yleensä kahteen (Katso Nintendo Entertainment System). Yleensä tällä aikakaudella tasohyppelypeleissä toimintoina olivat pelkästään hyppy ja jokin muu esim. ampuminen tai jokin muu vastaava ”erikoistaito”. Käytännössä suunnitelmallisesti tämän aikakauden tasoloikat ovat täysin toteutettavissa myös mobiilialustalla, kun taas 16-bittisen aikakauden pelit (4 toimintanäppäintä käytössä) voivat osoittautua hankalaksi ilman erillistä ohjainta.

Ehkä mobiilialustalla tarkoitus ei olekaan luoda konsolitasoloikan tuntumaa ja yrittää jäljittää konsolitasoloikan ohjattavuutta, vaan maksimoida mobiilialustan teknologian tuomat edut ja hyödyntää niitä innovatiivisen ohjattavuuden kautta. Vaikkei Robi Gangulyn artikkeli(Ganguly 2015) suoranaisesti liity pelinkehitykseen puhuu hän siinä kuitenkin siitä, miten monet mobiilisovelluskehittäjät tekevät virheen yrittäessään suoraan siirtää ”työpöytä” kokemusta mobiilialustalle, vaikka käyttäjän vaatimukset ja se, miten käyttäjä hyödyntää mobiilisovellusta ovat hyvin erilaiset.

Vaikka Shigeru Miyamoton vuonna 1985 kehittämän Super Mario Bros(Mariowiki 2015) pelin toiminnollisuudet oli rajattu teknologian puitteissa pelkkään hyppyyn ja horisontaaliin ohjattavuuteen, oli peli tehty muilla elementeillään todella vaihtelevaiseksi. Näitä elementtejä oli esimerkiksi hypyn korkeuden sääteleminen sen perusteella miten pitkään pelaaja painoi nappia pohjassa. Pelihahmo pystyy juoksemaan kun painaa nappia pohjassa, mutta pysähtyessään hahmon ohjaaminen hankaloituu. Hankaloituminen johtuu siitä, että pysähtyessään hahmo liukastelee hieman eikä pysähdy heti. Nämä elementit ovat yhä läsnä uusienkin Super Mario -pelien ohjauksessa.

4 Mobiilitasohyppelypelien konseptit ja niiden immersio

Immersion analysoinnissa ja tarkastelussa kohteena olivat samat pelin kuin kontrollianalyysissä. Keskityn tarkastelussa eri aspekteihin joilla pystyy luomaan peliin tunnelmaa, sekä immersiota. Immersiolla tarkoitetaan tässä yhteydessä sitä, miten pelaaja pystyy tuntemansa olevansa osa pelimaailmaa ja miten hyvin hän pystyy uppoutumaan pelikokemukseen sisälle.

Immersio liittyy pitkälti siihen, miten helposti pelaaja uppoutuu pelimaailmaan ja haluaa kokea olevansa osa sitä, eikä sen luomiseksi ole olemassa valmista keinovalikoimaa. Kuten Jamie Madigan artikkelissaan(Madigan 2010) kuvailee, että immersiivisessa pelissä pelaaja alkaa suosimaan median luomaa avaruutta sinä avaruutena missä pelaaja kokee tällä hetkellä ”olevansa”.

4.1 Konseptien analysointi

Tässä osiossa analysoin miten, viidessä eri suositussa, androidille tehdyssä mobiilitasohyppelypelissä on pyritty luomaan immersiota. Kiinnitän erityisesti huomiota äänimaailmoihin ja niiden vaihtelevuuteen sekä maailmojen syvyysasteisiin ja niiden vaihtelevuuksiin sekä tarinankerrontaan.

Vertailen lopuksi kyseisten pelien toteutusmenetelmiä meidän peliimme ja meidän toteutusmenetelmiin. Pyrin myös perustelemaan miksi meidän pelimme toteutusmenetelmät ovat sopineet paremmin meidän peliin, sekä miksi olemme pyrkineet tekemään erilaisia ratkaisuja.

4.1.1 Parallaksi

Parallaksi on kaksiulotteisissa peleissä käytettävä teknologia missä käytetty tausta liikkuu hitaammin suhteessa pelihahmon liikkuvuuteen(Gamesradar_US 2010). Kun pelaaja on liikkunut ruudulla matkan x, tausta liikkuu tästä huomattavasti pienemmän matkan. Tämä luo illuusion syvyydestä, vaikka syvyysakselia ei kaksiulotteisessa pelissä pystykään hyödyntämään(Odoerfer 2014).

Tämän tyylinen teknologia tuli tutuksi ensimmäistä kertaa valtavirralle pelissä Moon Patrol(Bogdan 2014, 118) ja Jungle Hunt(Odoerfer 2014) jonka jälkeen siitä on tullut hyvinkin yleinen elementti kaksiulotteisissa sivulta kuvatuissa tasohyppelypeleissä.

4.1.2 Maailman vaihtuvuus

Jos pelaaja etenee pelissä kenttien välissä. Vaihtuu pelin taustagrafiikat, sekä EPH:t, eli hahmot joita pelaaja ei pysty ohjaamaan. Myös muut visuaaliset voimavarat muuttuvat, luodakseen tuntua siitä, että pelaaja etenee uusiin paikkoihin pelissä.

Tällä pyritään luomaan immersiota ja vaihtelevuutta, jolla pyritään pitämään pelin mielenkiinto yllä. Vaikka pelattavuus olisikin samankaltaista jokaisessa kentässä, taustoja ja muita visuaalisia voimavaroja vaihtamalla voi mahdollisesti pitää pelaajan mielenkiintoa yllä.

4.1.3 Muut aspektit

Huomioin onko pelissä toteutettu minkäänlaista tarinankerrontaa, joko teksti tai ääninäyttelyn muodossa. Vaikka se olisikin hyvin vähäistä, huomioin sen kuitenkin, kunhan se on pelissä jossakin muodossa olemassa(Taulukko 2).

Otan myös huomioon taustamusiikin olemassaolon. Jos sitä ei ole niin analysoin minkä tyylisellä äänitoteutuksilla sitä on pyritty korvaamaan.

4.2 Immersion toteutus tarkastelluissa peleissä

Taulukko 2. Immersion parantaminen ja syvyyden luomisen tavat tarkastelluissa peleissä.

Pelin nimi	Parallaksi	Maailma vaihtuu	Tarinanker-rontaa	Taustamu-siikkia
Badland	Kyllä	Kyllä	Ei	Ei
Cordy 2	Kyllä	Kyllä	Kyllä	Kyllä
Granny Smith	Kyllä	Kyllä	Ei	Kyllä
Leo’s For-tune	Kyllä	Kyllä	Kyllä	Kyllä
Megatroid	Ei	Kyllä	Kyllä	Kyllä
Mikey Shorts	Kyllä	Kyllä	Kyllä	Kyllä
Murgorilla	Kyllä	Kyllä	Ei	Kyllä

Badland-pelissä immersiota luovat lähinnä vaihtuvat taustat jotka on toteutettu parallaksi-tekniikalla. Pelissä liikutaan vasemmalta oikealle eri kentissä ja taustalla on havaittavissa erilaisia otuksia ja metsikön näköistä maastoa. Pelissä ei ole tarinankerrontaa tekstimuodossa, vaan tarinan tapahtumakulun päättelemi-nen on jätetty täysin pelaajan varaan, kuten Joseph Leray toteaa arvostelus-saan(Leray 2013). Pelin äänimaailmaan ei kuulu musiikkia, vaan se on toteutettu tunnelmallisella taustamelulla, joka luo peliin hieman häiritsevän ilmapiirin.

Cordy 2 -pelissä immersiota on pyritty luomaan samoin kuin Badland-pelissä pa-rallaksi-tekniikalla luoduilla taustoilla, jotka vaihtuvat myös maailmojen välillä. Pe-lissä on myös luotu dialogia ja pyritty tuomaan muiden hahmojen kanssa interak-tiota ja tarinankerrontaa. Cordy 2 -pelissä äänimaailmassa hyödynnetään myös

taustamusiikkia, musiikki on iloisen kuuloista ja luo täten suhteellisen vaarattoman ilmapiirin pelille.

Granny Smith -peli keskittyy enemmän pelattavuuteen, mutta siinäkin hyödynnetään samaa kaavaa kuin aiemmissa peleissä, eli parallaksitaustaa joka vaihtuu maailmojen välillä. Parallaksitausta on toteutettu kolmiulotteisilla objekteilla, vaikka pelin etusijalla nähtävät objektit ovat kaksiulotteisia (hahmot). Myös pelin grafiikat muuttuvat kun pelin kenttiä läpäisee riittävästi ja pääsee pitemmälle pelissä. Minkäänlaista tarinankerrontaa pelissä ei ole toteutettu, eikä se toisi peliin kauheasti lisää, sillä peli on enemmän arcade-tyylinen muihin tässä listassa oleviin peleihin verrattuna. Kyseisessä pelissä on käytetty tunnelmanluojana taustamusiikkia. Taustalla soiva musiikki on suhteellisen neutraalia.

Leo’s Fortune -peli on kenties laajin pelikokemus kaikista näistä analysoiduista peleistä ja siinä ollaan nimenomaan keskitytty immersion luomiseen. Maisemat vaihtuvat kenttien sisällä hyvinkin paljon. Ensimmäisessä maailmassa liikutaan välillä niittymäisessä maastossa, kun taas välillä ollaan laavan täytteisessä luolassa. Taustojen vaihdokset on toteutettu erittäin sulavasti ja ne vaihtuvat useita kertoja yhden kentän sisällä. Pelin tarinankerronta on toteutettu maailmojen välillä, ja siinä kerrotaan hieman taustatarinaa siitä miksi pelin hahmo on ajautunut juuri kyseiselle alueelle.

Pelin äänimaailmaan on myös sijoitettu paljon resursseja. Pelihahmo puhuu itsekseen lähestyessään tiettyjä asioita pelissä, kuten ratkaistavia ongelmia jne. Pelissä käytetään useita musiikkikappaleita yhdessä kentässä ja pelin taustamusiikit vaihtelevat riippuen millaisessa maastossa pelaaja liikkuu. Pelin taustamusiikit vaihtelevat myös maailmojen välillä.

Megatroid-pelissä immersiota on pyritty luomaan tarinankerronnalla, sekä taustamusiikilla. Pelin grafiikat muuttuvat myös eri maailmojen välillä. Immersio ei liene tärkein aspekti kyseisessä pelissä, vaan tuntuu siltä, että on keskitytty enemmän pelattavuuteen ja sen hiomiseen.

Pelissä ei ole Granny Smith -pelin tyylistä maailmankarttaa, vaan pelissä edetään eteenpäin satunnaisesti generoiduissa kentissä ja kun on kerännyt tarpeeksi tasoja, pelaaja pääsee etenemään tarinassa eteenpäin, minkä jälkeen maailman ulkonäkö muuttuu.

Mikey Shorts -pelissä hyödynnetään myös parallaksitaustoja ja pelissä maailmojen välissä grafiikat muuttuvat. Pelissä on pyritty luomaan tarinankerrontaa, joka tapahtuu maailmojen välissä parilla lauseella. Se ei tunnu tosin kovin olennaiselta osalta kyseistä peliä.

Pelissä on pyritty luomaan vanhan koulukunnan tuntumaa äänimaailman avulla. Pelissä on musiikkia joka koostuu lyhyistä kokonaisuuksista jotka toistuvat kerta toisensa jälkeen.

4.3 Konseptianalyysin kiteytys ja vertaus Murgorilla-peliin

Kaikissa edellä mainituista peleissä oli jollain tavalla pyritty tekemään kokemuksesta immersiivinen, erilaisilla elementeillä. Testauksesta tuli kuitenkin tunne, että immersion luominen kehittäjän näkökulmasta ei ole ollut kovinkaan korkealla tärkeysjärjestyksessä. En koe, että pelikokemukseni olisi kärsinyt, tai peliin uppoutuminen olisi ollut yhtään hankalampaa, vaikka joistakin näistä peleistä olisi jätetty joitakin immersioon vaikuttavia aspekteja pois. Monissa testatuissa peleissä tuntuu, että painoarvo on pistetty lähestulkoon täysin pelattavuuteen ja visuaalisia voimavaroja vaihtamalla on vain pyritty luomaan hieman vaihtelevuutta kenttien välillä, eikä niinkään immersiivistä ”maailmaa” johon voi uppoutua.

Parhaiten immersiota oli pystytty mielestäni luomaan Leo’s Fortune -pelissä, joka oli mielestäni konsolitasoloikkaan verrattuna lähestulkoon verrattavissa konseptinsa laajuutensa puolesta. Myös Ryan Whitwam sanoo arvostelussaan (Whitwam 2014), että tehokkaalla monitasoisilla taustoilla ja etusijalle sijoitetulla grafiikalla peliin on pystytty luomaan todella vahvaa syvyyden tunnetta. Olen hänen kanssa samaa mieltä, pelissä on kiinnitetty kehityksessä huomiota juuri niihin asioihin, joihin kiinnitin huomiota suunnitellessani Murgorilla-peliä. Maailmat vaihtelivat mukavasti ja maailmojen sisälläkin oli erityyppisiä maastoja jotka poikkesivat

toisistaan paljon. Myös musiikki oli pelissä erittäin mielekästä kuunneltavaa. Kaikki nämä aspektit tekivät Leo’s Fortune -pelistä todella tasaisen ja immersiivisen kokemuksen kaikilta aspekteiltaan.

Huonoiten immersiota oli mielestäni pyritty luomaan Mikey Shorts -pelissä ja koen, että peli olisi kenties antanut itsestään viimeistellymmän kuvan ilman hätäisesti hoidetun tuntuista tarinankerrontaa ja geneerisiä maailmoja. Peli tosin loisti pelattavuudessaan, joka teki pelistä todella miellyttävän pelata, eikä immersion puute haitannut pelikokemusta.

Kuten aikaisemmin mainitsin monissa aikaisemmissa peleissä immersio ja konsepti eivät ole olleet todennäköisesti kovinkaan korkealla tärkeysjärjestyksessä, mikä taas Murgorilla-peliä suunnitellessa on ollut. Halusimme luoda niin äänien ja visuaalisten voimavarojen kautta immersiivisen kokemuksen joka yllättää pelaajan aina kun hän pääsee eteenpäin pelissä. Tämän olemme tehneet täysin siitä syystä, että koemme, että mobiilimarkkinoilla on tarvetta immersiivisille kokemuksille, joissa on laajat konseptit joihin voi uppoutua sisälle. Kuten niissä peleissä oli joita pelasimme esimerkiksi Playstation 1 –pelikonsolilla 90 luvulla.

5 Murgorilla-pelin kehityksessä tehdyt ratkaisut ja perustelut

Murgorilla-peli on tasohyppelypeli joka on suunniteltu mobiilialustoille (Android, iOS, WinPhone). Pelissä on pyritty ratkaisemaan konseptillisia ongelmia, mitä esiintyy mobiilipeleissä verraten esim. konsolitasohyppelypeleihin mitä pelasin nuorempana. Tarkoitus ei ole luoda konsolitasohyppelypeliä mobiilille, vaan ottaa konsolitasohyppelypelin olemus ja tehdä siitä mobiiliystävällinen käännös. Tämä oli pääasiallinen idea kun lähdimme kehittämään Murgorilla-peliä.

Pelin kehitys on käynyt lävitse lukuisia iteraatioita, jolloin pelin ydinmekaniikat ja suunnitelma ovat muuttuneet vahvasti. Olemme prototyyppejä tehdessä toden-

neet, etteivät tietyt aspektit toimi ja muuntaneet eri mekaniikkojen toiminnollisuutta kehityskaaren edetessä. Kuten esimerkiksi aikaisemmin kontrolliosuudessa mainitsemani virtuaalisilla kontrolleilla toteutettu tähtäys, joka ei sopinut pelin teemaan, sekä muita vähemmän merkittäviä pelisuunnitteluun liittyviä aspekteja.

5.1 Idean syntyperä

Pelin alkeellinen kehitys alkoi vuonna 2013, kun saimme Aki Kupiaisen kanssa idean alkaa kehittämään mobiilipeliä, nähtyämme Fingersoft-yrityksen kehittämän Hill Climb Racing -nimisen pelin jymymenestyksen. Jos he menstyivät heidän pelillään, miksi emme me voisi menestyä?

Kävimme yhdessä Aki Kupiaisen kanssa Game Programming Basics -nimistä kurssia Joensuun Karelia ammattikorkeakoulussa, jossa tarkoituksena oli tehdä peli Unity 3D(Unity Technologies 2015a) -pelimoottorilla. Tämän kurssin aikana keräsimme suhteellisen paljon tietämystä ja osaamista siitä, miten perusasiat toimivat kyseisessä pelimoottorissa ja päätimme, että tällä on hyvä lähteä kehittämään, sen enempää asiaa tutkimatta. Olimme myös saaneet selville, että Unity 3D –pelimoottorin ilmaisversiolla pystyy hankkimaan 100 000 €, ennen kuin tarvitsee alkaa maksamaan lisenssimaksuja Unity Technologies -yritykselle.

5.2 Työkalujen valinta

Tässä osiossa tuon esille meidän käyttämämme työkalut ja pyrin perustelemaan miksi valitsimme kyseiset työkalut tämän projektin työstämiseen. Keskityn lähinnä toteutukseen liittyviin työkaluihin, enkä niinkään hallinnollisiin.

Perustelen meidän pelimoottorin valintaa ja mitä liitännäisiä ja muita sovelluksia käytimme projektin toteuttamiseen. Lukijana on hyvä huomioida, että seuraava

osio viittaa aikaan jolloin aloitimme kehittämään projektia. Tämä ajankohta sijoittuu vuoteen 2013, joten nykytilanteessa olisimme voineet tehdä erilaiset ratkaisut ja siitä tulee enemmän tietoa myöhemmin.

5.2.1 Pelimoottori

Siihen aikaan kun aloimme kehittämään Murgorilla-peliä, muut pelimoottorit olivat Game Maker(YoYo Games 2015a) -pelimoottoria lukuun ottamatta, kaksiulotteisilta ominaisuuksiltaan yhtä rajallisia kuin Unity 3D -pelimoottori. Game Maker -pelimoottori ei meidän kehityksen alkaessa tukenut mobiilialustoille kehittämistä, joten sekin piti jättää pois pelimoottorin valinnasta(Pelin ensimmäinen prototyyppi on tosin tehty kyseisellä pelimoottorilla pc:lle). Mietimme myös hetken LibGDX-nimistä java-ohjelmointikieleen pohjautuvaa pelimoottoria, joka olisi ollut täysin ilmainen. Emme kuitenkaan ottaneet sitä käyttöön rajatun graafisen käyttöliittymän vuoksi.

Unreal Engine- ja Cry Engine –pelimoottorit oli lähestulkoon täysin suunniteltu tällä hetkellä kolmiulotteisten pelien kehittämiseen ja niiden hinnoittelu oli suunnattu enemmänkin isoille pelitaloille. Tästä syystä päädyimme lopulta Unity 3D -pelimoottoriin, joka oli meidän tarpeisiin ja käyttömukavuudeltaan tähän projektiin mieluisin.

Unity 3D -pelimoottori tarjoaa edullisen ja sulavan käyttöliittymän pelimoottorille, joka taipuu lähestulkoon kaikkiin haluttuihin muotoihin, ilman, että lähdekoodiin tarvitsee perehtyä. Unity 3D -pelimoottorissa peliobjektit lisätään nk. näyttämölle(Scene), minkä jälkeen niihin lisätään komponentteja. Eri komponentit toteuttavat erilaisia toimintoja ja niiden lisääminen on tehty helpoksi.

Koodia voi kirjoittaa kolmessa eri ohjelmointikielessä, jotka ovat C#, Boo, sekä Java Script(Ei tosin täysin puhdasta Java Scriptiä). Kirjoitettu koodi lisätään samalla tavalla, kuin mikä tahansa muu komponentti peliobjektiin. Koodissa voi periä eri asioita Unity 3D -pelimoottorin sisäisistä luokista, jotka toteuttavat erilaisia haluttuja toimintoja peliobjekteille.

5.2.2 Liitännäiset ja muut työkalut

Kuten aiemmin mainitsin, Unity 3D -pelimoottorin kaksiulotteiset ominaisuudet olivat rajalliset kun aloimme peliä kehittämään. Tästä syystä jouduimme käyttämään liitännäisiä, jotta pystyimme simuloimaan kaksiulotteisia asioita Unity 3D

-pelimoottorin kehitysympäristössä. Tähän valitsimme 2D Toolkit -työkalut, jolla pystyi implementoimaan nk. Sprite-kuvia Unity 3D -pelimoottoriin ja niistä pystyi tekemään animaatioita.

Käytimme myös Master Audio -nimistä liitännäistä äänten organisoimiseen pelimoottorin sisällä. Tämän liitännäisen avulla musiikkikappaleiden vaihtaminen kenttien sisällä oli tehty helpoksi liitännäisen sisäisillä funktioilla. Meillä on ensimmäisessä kentässä pelissä jo kolme musiikkikappaletta, joten tämä oli erittäin tärkeä ominaisuus meille.

5.3 Murgorilla-pelin kontrollit

Kun aloimme kehittämään Murgorilla-peliä Unity 3D -pelimoottorin sisäänrakennetuilla fysiikoilla oli hankala toteuttaa kaksiulotteisen tasohyppelypelin reaktioherkkää fysiikkaa, sillä kolmiulotteiset fysiikat Unity 3D -pelimoottorissa hakevat nk. realistista fysiikkaa ja pyrkivät imitoimaan maapallon fysiikkaa mahdollisimman tarkasti. Tasohyppelypeleissä nk. realistiset fysiikat ei tosin ole useimmissa tilanteissa haluttua, koska jos pelaaja on nk. ”liian kelluva”, hahmon liikkuminen voi tuntua pelaajasta vähemmän herkältä ja tästä syystä tehdä pelikokemuksesta epämiellyttävän. Tästä syystä päätimme ohjelmoida hahmolle omat fysiikat, joilla pystyy säätämään painovoimaa, jotta pelaaja saavuttaisi tason pinnan nopeammin, eikä vaikuttaisi kelluvalta.

Toteutuksemme tarvitsisi muuttujan joka määrittää painovoiman, sekä tyhjän vektorin, jolla pystymme määrittämään mihin suuntaan pelaaja on kulkemassa. Alla on funktio(Koodilistaus 1.) joka hoitaa pelaajan fysiikkojen simuloimisen näyttämöllä. _moveDirection on Vector3-tyyppinen muuttuja josta lasketaan mihin pelaajahahmoa halutaan liikuttaa kentällä, riippuen siitä mikä pelaajan tila on tällä hetkellä.

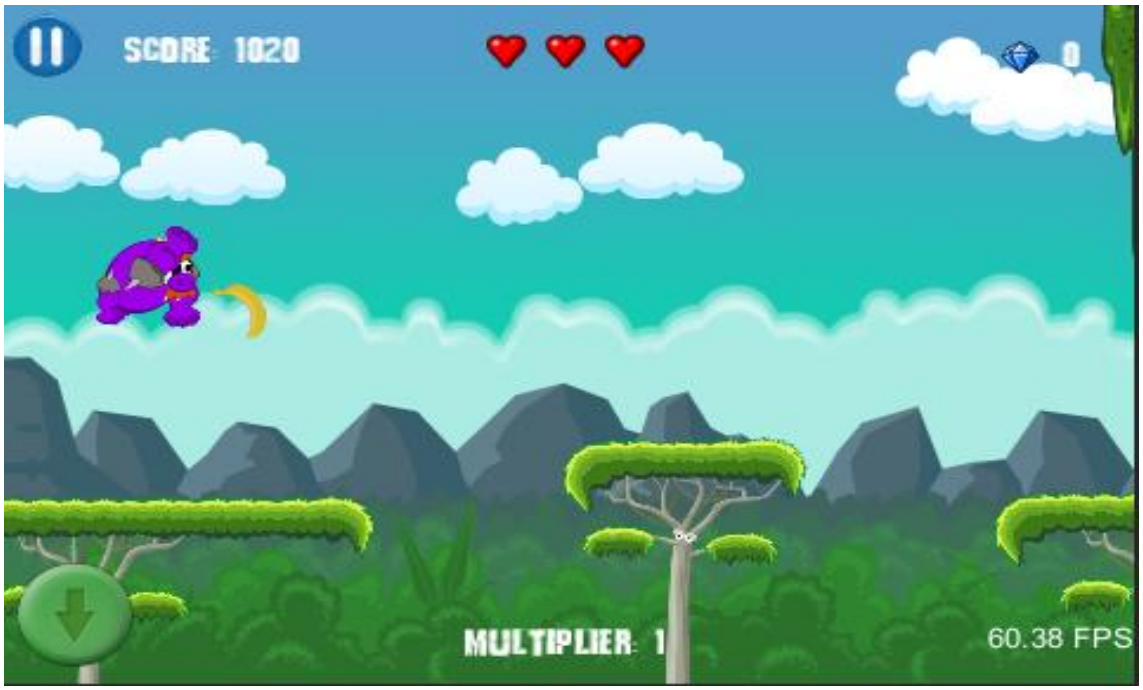
```
void MovePlayer()
    {
        _moveDirection.y -= _gravity * Time.deltaTime;

        if (_moveDirection.y <= _jumpSpeed - _gravity && GameManager.Instance.Cur-
rentMovementState != PlayerMovementState.Diving)
        {
```

```
    _moveDirection.y = _jumpSpeed - _gravity;
}

if (_moveDirection.y <= -_diveSpeed)
{
    _moveDirection.y = -_diveSpeed;
}

_controller.Move(_moveDirection * Time.deltaTime);
    }
```

Koodilistaus 1. Funktio jossa simuloimme painovoiman pelihahmolle, sekä tarkastelemme onko hahmo syöksymässä.

5.3.1 Ampumisen toteutus

Ampumisen ja muiden toimintojen toteuttaminen piti saada tehtyä niin, että puolet näytöstä reagoi eri tavalla kuin toinen puoli näytöstä. Unity 3D -pelimoottorin Screen-luokasta löytyy pelkästään luettava muuttuja nimeltä ”width” josta pystyy lukemaan näytön leveyden. Touch-luokan kautta pääsee käsiksi kosketuksen positioon, pystymme ehtolauseen avulla katsomaan onko kosketuksen positio jommallakummalla puolella näyttöä x-akselilla kun jaamme Screen.width muuttujan kahdella. Tällä tavalla pystymme ottamaan syöttöä molemmilta puolilta näyttöä samanaikaisesti, mikä mahdollistaa useiden toimintojen tapahtumisen samanaikaisesti. Ei myöskään käy niin, että kun pelaaja haluaa esimerkiksi ampua, niin hahmo myös hyppää samalla hetkellä. Ampuminen tapahtuu painaessa ruudun oikeaa reunaa, jolloin pelaajahahmo heittää projektiilin matkaan(Kuva 11).



Kuva 11. Kuva pelistä kun pelaajahahmo heittää projektiilin matkaansa.

Alla on pala koodia(Koodilistaus 2), miten kyseisen asian pystyy toteuttamaan Unity 3D -pelimoottorissa. Luodaan kosketustyyppinen(Touch) taulukko, johon laitetaan Input luokasta saatavilla olevat kosketukset(Input.touches) ja niitä pyöritellään for-silmukassa. Kosketus luokasta löytyy enum-tyyppinen muuttuja nimeltä phase, jolla pystyy tarkkailemaan missä tilassa kyseinen kosketus on tällä hetkellä, mikä auttaa hallinnoimaan kosketustoimintoja.

```
for (int i = 0; i < Input.touchCount; i++)
        {
            Touch[] _touch = Input.touches;

            if (_touch[i].position.x < Screen.width * .5f && _touch[i].position.y
< Screen.height && _controller._isGrounded && _touch[i].phase == TouchPhase.Began)
            {
                _moveDirection.y = _jumpSpeed;
                GameManager.Instance.CurrentMovementState     =     PlayerMovement-
State.Jumping;
            }

                                        if (_touch[i].position.x < Screen.width * .5f
&&   _touch[i].position.y   <   Screen.height   &&   !_controller._isGrounded   &&
_touch[i].phase == TouchPhase.Began)
            {
                _moveDirection.y = -_diveSpeed;
                GameManager.Instance.CurrentMovementState     =     PlayerMovement-
State.Diving;
            }
```

```
            if (_touch[i].position.x > Screen.width * .5f && _touch[i].phase ==
TouchPhase.Began)
            {
                StartCoroutine(Shoot());
            }
        }
```

Koodilistaus 2. Funktiossa tarkastellaan pelaajan kosketuspainalluksia ja toteutetaan eri toimintoja niiden perusteella.

Emme käyttäneet virtuaalisia painikkeita ampumisen tai muiden kontrollien toteuttamiseen, koska lopullisessa versiossa toimintojen määrä oli niin vähäinen, että niille ei ilmennyt tarvetta. Karsiessamme pois virtuaaliset painikkeet pystyimme luomaan selkeän ohjauksen ja minimoimaan virhepainalluksien määrän, mikä saattaisi pilata pelikokemuksen näin nopeatempoisessa pelissä.

5.3.2 Vihollisten murskaaminen syöksymällä ja sen toteuttaminen

Pelin toinen mekaniikka vihollisten tuhoamiseen on se, että pelaaja syöksyessään vihollisen päälle tuhoaa vihollisen ja sinkoaa takaisin ilmaan. Tässä oleellisinta oli saada jollain tavalla luettua, että pelaaja on vihollisen yläpuolella kun kosketus viholliseen tapahtuu. Tämän jälkeen on helppo määrittää mitä tässä tilanteessa tehdään. Pelaajan on myös tarkoitus ottaa vahinkoa jos hän osuu viholliseen ilman, että lähestyy vihollista ylhäältäpäin. Tämä mekaniikka on tuttu useista tasohyppelypeleistä joista legendaarisimmat lienee Super Mario Bros, sekä Ducktales Nintendo(Rignall 1990) Entertainment System pelit pelikonsolilla.

Ohjelmoinnin osalta vihollisten murskaaminen syöksymällä on toteutettu sillä tavalla, että katsotaan pelaajan positio y-akselilla ja verrataan sitä vihollisen positioon. Pitää myös huomioida pelaajan positio x-akselilla. Tämä on tärkeää välttääkseen tilanteen jossa pelaaja hieman hipaisee vihollisen törmäyttimen kulmaan, joka tekisi mekaniikasta liian helpon suorittaa.

```
void OnTriggerEnter(Collider _col)
{

if (_player.transform.position.y - _colliderSizeY >= transform.position.y &&
_col.gameObject.tag == _player.tag && GameManager.Instance.CurrentMovementState ==
PlayerMovementState.Diving)
```

```
            {
            PcControls._moveDirection.y = 10;
            PlayerControls._moveDirection.y = 10;
            PoolManager.SpawnPools["Misc"].Spawn("EnemyExplosion",    new    Vec-
            tor3(transform.position.x,   transform.position.y,   -10f),   Quater-
            nion.identity);
            _fsm.ChangeState(JumpingSlimeStates.Reset);
            }

}
```

Koodilistaus 3. Esimerkki Unity3D OnTriggerEnter funktiosta ja sen hyödyntämistä Murgorilla-pelissä.

Pelaajalla ja vihollisella siis Collider-komponentti joka katsoo tarkastaa osumat pelaajan ja vihollisen välillä. Ne ovat asetettu laukaisin(Trigger) muotoon. Koska jos ne olisivat normaalitilassa, objektit eivät pystyisi liikkumaan toistensa lävitse halutessaan. Kun pelaaja osuu viholliseen yllä oleva metodi(OnTriggerEnter) kutsutaan, jossa toteutetaan halutut toiminnot. Tässä tapauksessa toteutamme ehtolauseen, jossa tarkastamme milloin pelaaja ja vihollinen törmää toisiinsa. Kun tämä tapahtuu tuhoamme vihollisen ja pelaaja hyppää takaisin ilmoihin.

5.4 Nykyiset kontrollit ja niihin päätyminen

Murgorilla-pelin ominaisuudet olivat alun perin seuraavat(Toteutusmenetelmä 1):

1. Hyppy, joka toimi kun pelaaja painoi vasenta puoliskoa ruudusta, eli 50 prosenttia ruudun koosta.

2. Sukellus, joka aktivoitui samalta alueelta kuin, hyppy, mutta vasta kun pelaaja oli ilmassa. Sukellus tarkoittaa käytännössä sitä, että pelaaja voi ohjata hahmon missä tahansa tilanteessa laskeutumaan välittömästi sen alla olevalle tasolle.

3. Ampuminen, joka toimi painamalla ruutua siitä kohdasta mihin halusi ammuksen lähtevän.

Kun testasimme näitä ominaisuuksia 3,5” -näyttöisellä puhelimella, asia toimi todella mallikkaasti ja peli oli täysin pelattava. Kun saimme vihdoin testikäyttöön

ruudultaan suurempia laitteita, huomasimme, että ampuminen on todella hankalaa, eikä lainkaan hauskaa. Tämän jälkeen lähdimme soveltamaan virtuaalista ”analogista tattia” ruudulle. Pienen harjoittelun jälkeen kontrollitapa tuntui todella toimivalta ja olimme täysin varmoja päässeemme täydelliseen lopputulokseen(Toteutusmenetelmä 2). Annoimme kuitenkin pelimme testattavaksi ja mekaniikka osoittautui aivan liian vaikeaksi. Sen luoma ”tarkkuus” elementti ei oikein sopinut muuten nopeatempoiseen reaktiopohjaiseen peliin. Tästä syystä päätimme luopua noin vuoden kehityksen jälkeen tästä ominaisuudesta ja tehdä siitä ”järkevämmän”. Tämänhetkisessä suunnitelmassa on seuraavat ominaisuudet:

1. Hyppy, joka toimi kun pelaaja painoi vasenta puoliskoa ruudusta, eli 50 prosenttia ruudun koosta.

2. Sukellus, joka aktivoitui samalta alueelta kuin, hyppy, mutta vasta kun pelaaja oli ilmassa. Sukellus tarkoittaa käytännössä sitä, että pelaaja voi ohjata hahmon missä tahansa tilanteessa laskeutumaan välittömästi sen alla olevalle tasolle.

3. Aseen heittäminen ruudun toiseen reunaan, jos ase osuu viholliseen se kääntyy takaisin ja palaa pelaajan hallintaan.

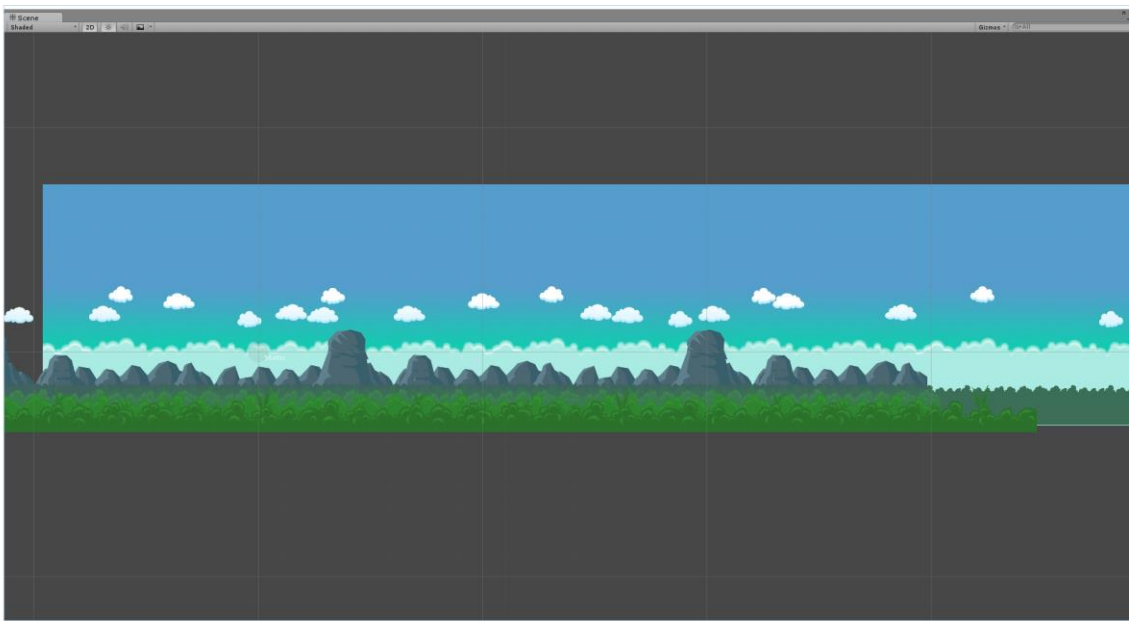
4. Pelaaja pystyy syöksyessään tuhoamaan vihollisen syöksymällä tämän päälle. Pelaaja hyppää myös takaisin ilmaan, kun näin tapahtuu.

Tätä mekaniikkaa(Toteutusmenetelmä 3) on jo testattu suhteellisen paljon ja olemme tulleet lopputulokseen, että se vaikuttaa paljon luonnollisemmalta/hauskemmalta mekaniikalta, kuin aikaisempi tarkkuutta vaatinut ampumismekaniikka. Vaikka toiminnollisuuksia on kolmen sijaan neljä, niiden toteuttaminen kosketusnäytöllä on paljon sulavampaa ja ei vaadi minkäänlaisia virtuaalisia painikkeita. Emme tarvinneet toteuttamiseen erityistä tekniikkaa ja pystyimme samalla ruudunjaolla toteuttamaan enemmän toimintoja kuin aikaisemmin. Virtuaalisista painikkeista luopuminen mahdollisti monipuolisemman pelikokemuksen selkeämmällä käyttöliittymällä.

5.5 Parallaksitaustan toteutus

Toteutimme parallaksin peliin ensin tekemällä grafiikkaa joka on jaettu useampaan tasoon. Tämän jälkeen liikutamme eri tasoja eri vauhdilla sen mukaan, mikä niiden sijainti on kolmiulotteisen vektorin z-akselilla. Mitä kauempana grafiikka on kamerasta, sitä hitaammin liikutamme grafiikkaa kolmiulotteisen vektorin x-akselilla. Mitä lähempänä grafiikka on kamerasta, sitä nopeammin liikutamme sitä. Tällä pystymme luomaan illuusion siitä, että kaksiulotteisessa pelissä on syvyyttä ja pelaajalle välittyy kokemus, että pelihahmon maailma sisältäisi myös kolmannen ulottuvuuden.

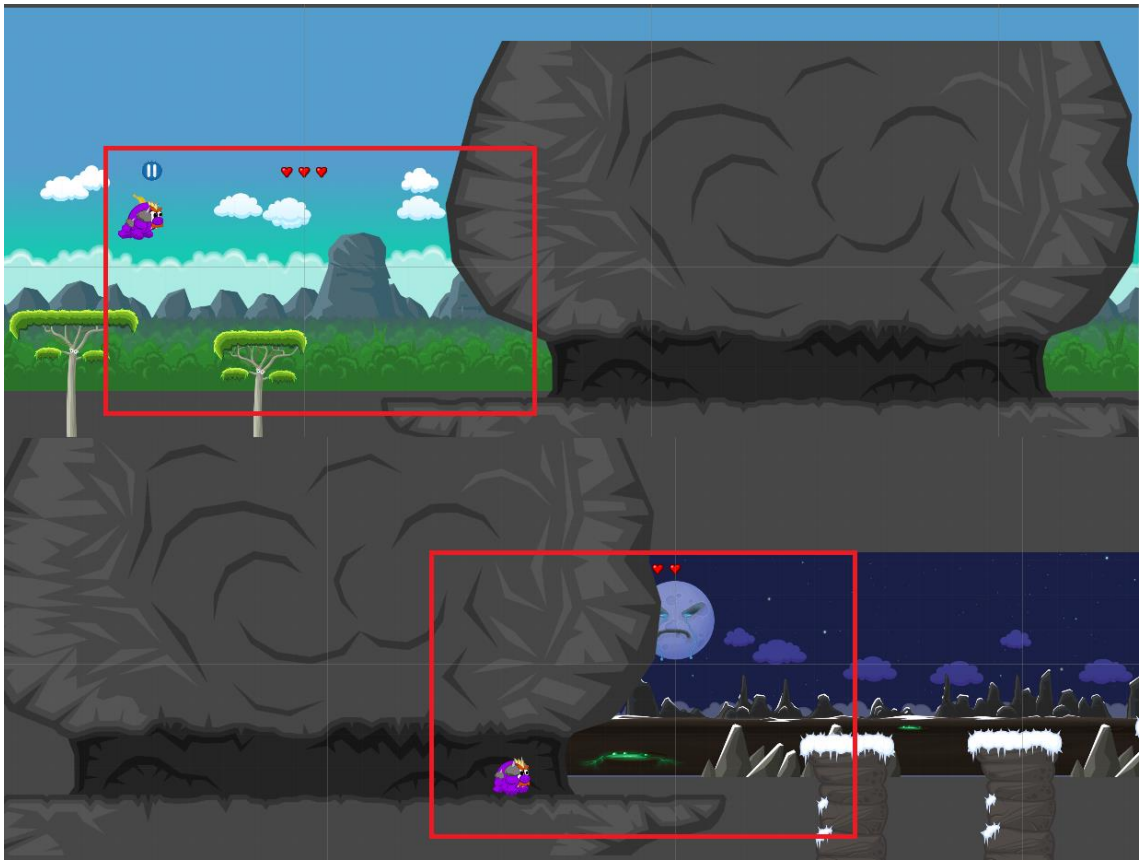
Jokainen grafiikkapalasen pituus on määritetty ennakkoon ja se on standardisoitu niin, että kaikki palaset ovat yhtä pitkiä. Palasia tuodaan ruudulle jonossa joka liikkuu x-akselilla ja niitä on samanaikaisesti ruudulla maksimissaan 2(Kuva 7). Kun yksi palanen on poistunut kokonaan ruudulta, seuraava ilmestyy viimeisimmän taakse, jonka jälkeen ruudusta poistunut palanen poistetaan, jotta näyttämölle ei jää liikaa peliobjekteja.



Kuva 7. Murgorilla-pelin näyttämöltä joka näyttää miten parallaksi toimii.

5.6 Kentän vaihtaminen lennosta

Koska pelissämme ei käytännössä tule taukoja pelikenttien välissä, vaan se on yhtä jatkumoa, meidän piti keksiä menetelmä jolla pystymme vaihtamaan pelaajan huomaamatta näitä parallaksitaustan palasia. Tähän ratkaisumme oli tuoda koko näytön peittävä ontto kivi jossa pelaaja pääsee hetkeksi juoksemaan(Kuva 8). Kun pelaaja juoksee kiven sisällä, pystymme vaihtamaan kiven takana olevat taustat seuraavan kentän taustoihin, mikä luo illuusion, että pelaaja on edennyt seuraavaan maailmaan(Kuva 9).



Kuva 8. Demonstroidaan näyttämöllä miten tason vaihto toimii suuren kalliografiikan kohdalla.

Kuva 9. Kuva pelaajan näkymästä samalla hetkellä kun kenttä vaihtuu.

Päädyimme tähän menetelmään sen takia, että se oli hyvin helppo toteuttaa ja saimme ratkaisun vielä todella kevyeksi koodin suhteen. Meillä oli myös mietteillä ratkaisuja, jossa olisimme esim. sumentaneet taustat toisiinsa, tai tehneet vaihtoehtoisia taustoille nk. ”vaihdospaloja” jotka tuotaisiin pelikentälle, kun haluaisimme vaihtaa kenttää. Nämä toteutukset tuntuivat erittäin hankalilta ja aiheuttaisi paljon lisätöitä grafiikan kanssa, kun taas ensimmäisessä kertomassani menetelmässä tarvitsisimme vain yhden palasen grafiikkaa. Kokeilimme ensimmäistä toteutustapaa ja tulimme lopputulokseen, että se näyttää erittäin hyvältä ja luo pelaajalle todella suuren yllätyksen, koska luolan aikana maisema on muuttunut täysin.

Kuten aikaisemmin mainitsin, Unity 3D -pelimoottorin kaksiulotteiset ominaisuudet olivat erittäin heikot kun aloimme kehittämään Murgorilla-peliä. Tästä syystä meidän piti hankkia ulkopuolinen lisäosa Unity 3D -pelimoottoriin, jolla pystyi toteuttamaan kaksiulotteisia ominaisuuksia. Tähän valitsimme 2D Toolkit -nimisen lisäosan. Lisäosa jakaa kaiken käytössä olevan grafiikan Sprite ID -nimisen muuttujan mukaan ja jos tämän asettaa koodissa jollekin objektille eriksi kuin mitä se aikaisemmin oli, määrittyy objektin näytettävä kuva uudestaan. Nykyään Unity 3D

-pelimoottorissa on natiivina lisätty kaksiulotteisia ominaisuuksia ja saman asian pystyisi nykyään toteuttamaan ilman ladattavaa lisäosaa.

6 Pohdinta

Tässä osiossa pyrin tuomaan esille johtopäätöksiä ja muita näkökulmia opinnäytetyön aikana tehdystä prosessista. Kiteytän huomiomani asiat kontrollien toteuttamisesta sekä immersion luomisesta mobiilitasohyppelypelissä, sekä tuon esille mietteitä miten asiat olisi voinut toteuttaa eritavalla kuin me toteutimme.

Tuon myös esille eri vaihtoehtoja nykytilanteessa pelimoottorille ja työkaluille aloittelevalle pelinkehittäjälle suotuisasta näkökulmasta. Kiinnitän erityisesti huomiota ilman liitännäisiä toteutettaviin kaksiulotteisiin ominaisuuksiin, sekä pelimoottorin hankintahintaa.

6.1 Murgorilla-pelin immersion kehitys

Kun lähdimme kehittämään Murgorilla-peliä, halusin suunnittelussa ottaa erityisesti huomioon, miten mahdollisesti voisi tuoda samanlaisen tasohyppelykokemuksen johon itse totuin lapsena, nykyajan lapselle mobiilialustalla. Lapsuudessa/nuoruudessani pelaamissani seikkailu/tasohyppelypeleissä oli todella paljon syvyyttä maailmallisesti ja tarinallisesti. Seikkailtiin useissa erilaisissa ympäristöissä ja musiikit vaihtelivat ympäristön mukaan sopiviksi. Todella useiden lasten ainut kosketus videopeleihin tulee mobiilipelien kautta ja todella useista peleistä puuttuu vastaavanlainen syvyys kuin konsolitasohyppelypeleistä joihin minä totuin lapsena(Katso Crash Bandicoot -peli, Klonoa: A door to panthomile -peli, Jak and Daxter -pelisarja)

Murgorilla-pelissä tämä ilmenee, siinä että halusin luoda useita mielenkiintoisia maailmoja peliin ja seikkailun tuntua simuloivan tarinallisen jatkumon. Halusin

myös luoda pelaajalle tunteen, että hän etenee syvemmälle pelimaailman syövereihin. Vaikka toistaiseksi konsepti rajoittuukin vain viiteen maailmaan, niin niiden väliset erot ovat suuria ja tällä saadaan pelaajalle vietyä kuva elävästä maailmasta.

Tarina on todella haastava aihe, kun puhutaan mobiilipelistä. Usein alussa esitettävä tarinankerrontaa sisältävän ”väli-animaation” jätän katsomatta, koska haluan suoraan pelaamaan. Pelissämme joka perustuu kuitenkin enemmän pelattavuuteen, koemme kuitenkin, että on tärkeää ettei pelaajaa ylikuormiteta heti kättelyssä ja pelin tarina tuodaan pelaajalle, enemmänkin alitajuisesti. Tällä tarkoitan sitä, että eri asiat pelimaailmassa viestivät pelaajalle enemmän, kuin että tarina tuotaisiin esim. tekstimuodossa suoraan pelaajan luettavaksi. Tästä syystä suunnitelmana on implementoida pelivalikkoon painike josta pelaaja voi halutessaan päästä näkemään pelin tarinaa ja jos aika ja resurssit riittävät, olemme ajatelleet tekevämme todella lyhyitä pelaajan halutessaan ohitettavia väli-animaatioita, joissa pelin tarinaa viedään eteenpäin kenttien välissä.

Alun perin peliin oli tulossa 2 pelattavaa hahmoa, millä halusimme tuoda niin tytöille kuin pojille helposti samaistuttavan hahmon. Pelissä ollut määrätietoinen taitava noitanainen olisi toiminut tytöille mieluisana hahmona ja nk. esikuvana, kun taas pojille toimi vielä pelissä oleva gorillanomainen hahmo, joka on vahva ja nk. ”viileä” hahmo, jolle pelaaja pystyy hankkimaan uusia varusteita. Tämän tyttöhahmon sovittaminen ruudulle oli todella vaikeaa, koska peli pyörii todella pienellä näytöllä, joten poistimme hänet toistaiseksi pelistä ja kehitimme vain feminiinin version gorillanomaisesta hahmosta. Pelaaja saa pelin alussa valita naaras/uroshahmosta. Tämä ei niinkään viittaa konseptin syvyyteen itsessään, vaan lähinnä siihen kokemukseen/viestiin jonka haluamme pelaajan kokevan/saavan.

Kysymys jonka voi esittää on, voiko pelin konsepti olla yhtä laaja niin mobiili, kuin konsolipelissä, vai pitääkö joitain asioita karsia pois? Koimme, että pelin tulisi mekaniikoiltaan olla mahdollisesti todella paljon simppelimpi saadakseen kontrollit toimimaan mobiilialustalla. Kokemus jonka haluamme pelaajalle luoda voi olla lähestulkoon yhtä laaja kuin konsolitasoloikassa. Tähän ainakin pyrimme meidän pelikonseptillamme.

6.2 Muut mahdolliset pelimoottorit

Kuten aikaisemmin ilmeni, käytimme Unity 3D -pelimoottoria Murgorilla-pelin toteuttamiseen. Kehitys aloitettiin yli puolitoista vuotta sitten, jolloin Unity 3D -pelimoottorin kaksiulotteiset ominaisuudet ja mobiilikehitysominaisuudet olivat rajoitetut ja lukuisten muiden pelimoottoreiden myös. Nykyään tilanne on erilainen ja mahdollisuuksia on enemmän.

6.2.1 Unity 3D 5

Unity 3D 5 on Unity Technologies -nimisen yrityksen kehittämä pelimoottori. Sillä pystyy täysivaltaisesti kehittämään, niin kaksiulotteisia kuin myös kolmiulotteisiakin pelejä. Pelimoottori tukee laajalti kaikkia yleisimpiä alustoja, joille pelejä pystyy kehittämään, niin HTML 5 alustasta Playstation 4 alustaan. Käytännössä pelin tarvitsee ohjelmoida vain kertaalleen(pieniä hienosäätöjä lukuun ottamatta) pystyäksesi julkaisemaan lukuisille eri alustoille. Kehityksen aloittaminen on täysin ilmaista jos tyydyt siihen, että peliisi jää Unity-vesileima, sekä 100 000 $ tulokatto(Unity Technologies 2015b).

Pro versio maksaa tällä hetkellä 1500€ kertamaksuna tai 75/kk jos rekisteröityy Unity 3D 5 pro –pelimoottorin kuukausitilaajaksi(Unity Technologies 2015c). Jos ryhtyy tilaajaksi, sitoo tämä kuitenkin vuoden vähittäistilaukseen, joten vuoden tilaus maksaa kokonaisuudessaan 900 euroa. Pitää kuitenkin huomioida, että vaatimus kehittämiselle, on yksi Unity 3D 5 lisenssi jokaista Unity 3D 5 pelimoottoria käyttävää kohden tiimissä. Joten jos kolme henkilöä kehittää peliä ja haluaa julkaista Pro versiolla, pitää kaikilla olla aktiivinen lisenssi käytössään. Myöskään Free ja Pro lisenssejä ei saa sekoitella keskenään(Unity Technologies b 2015).

Unity 3D 5 -pelimoottorin kaksiulotteiset ominaisuudet on paljon laajemmat, kun mitä ne olivat kun aloitimme aikoinaan Murgorilla-pelin kehityksen. Nykyään pelimoottori perusominaisuuksiin sisältyy kaksiulotteisia ominaisuuksia, ilman lisäkustannuksilla hankittuja liitännäisiä. Liitännäisillä pystyy tosin hankkimaan omi-

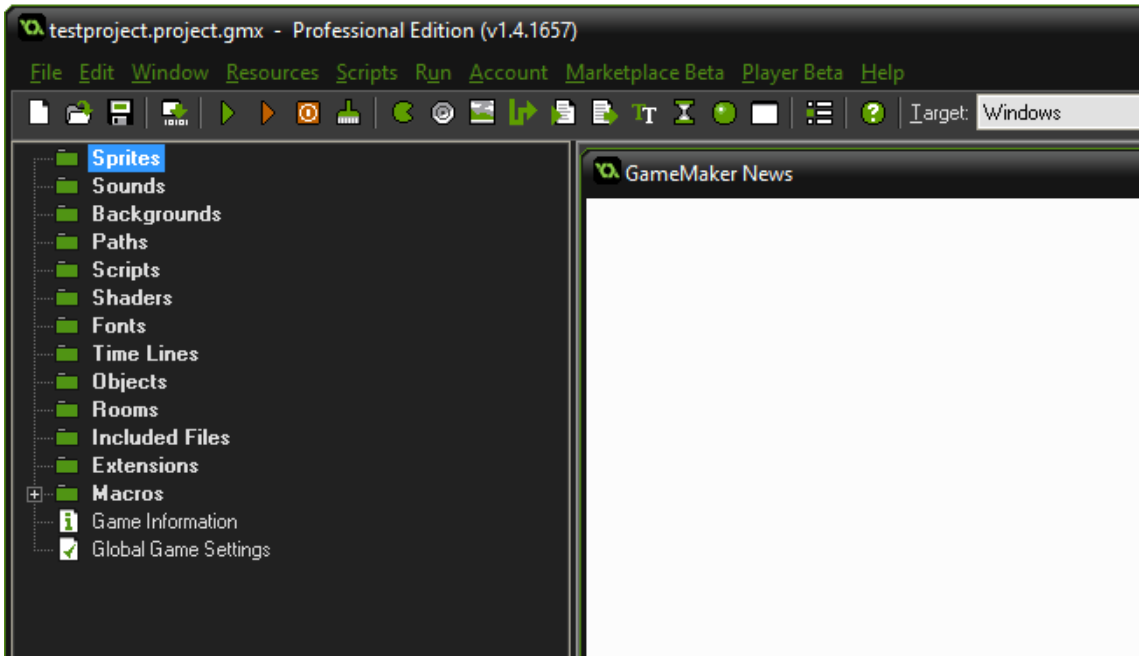
naisuuksia, joita voi olla hankala ohjelmoida aloittelijana ja jotka voivat luoda optimoidumman kokemuksen. Myös mobiilialustoihin relevanttien mainoksien tai mikromaksujen implementoimiseen on paljon asiaa helpottavia liitännäisiä kuten esimerkiksi Android Native Plugin –liitännäinen(Stan’s Assets 2015).

6.2.2 Game Maker Studio

Game Maker Studio on YoYo Games -yrityksen kehittämä pelimoottori, joka keskittyy lähinnä täysin kaksiuloitteisten pelien tekoon. Kun aloitimme Murgorilla-pelin kehityksen, Game Maker Studio -pelimoottorilla ei pystynyt kehittämään mobiilialustoille. Nykyään tuki on lisätty kaikille yleisille mobiilialustoille(iOS, Android, Windows Phone ja Blackberry), joten peli olisi ollut mahdollista toteuttaa julkaisukelpoiseen kuntoon myös Game Maker Studio pelimoottorilla.

Game Maker studion käyttöliittymä on hieman vanhentunut, eikä se tue yleisiä ohjelmointikieliä vaan käyttää omaa ohjelmointikieltään nimeltä Game Maker Language -ohjelmointikieltä, joka on yksinkertaistettu versio perinteisistä ohjelmointikielistä. Alla on vielä esimerkki Game Maker Language -ohjelmointikielellä toteutetusta silmukasta, sekä käyttöliittymästä(Kuva 10).

```
var i;
i = 0;
repeat(10)
{
    i += 1
}
```



Kuva 10. Kuva Game Maker Studio -pelimoottorin käyttöliittymästä

Game Maker Studio -pelimoottorin hinta on 149 $, sekä 299$ per mobiilialusta jolle haluaa julkaista(YoYo Games 2015b). Pelimoottorin voi myös ostaa pakettihintaan joka on 799$(YoYo Games c 2015c). Tämä sisältää julkaisun kaikille mahdollisille alustoille, sekä tulevaisuudessa tuetut kehitysalustat.

6.2.3 Unreal Engine 4

Unreal Engine 4 -pelimoottorin kaksiulotteiset ominaisuudet toteutetaan Paper 2D -nimisellä liitännäisellä joka on nativisti implementoituna Unreal Engine 4 -pelimoottoriin tällä hetkellä. Sillä pystyy toteuttamaan kaksiulotteisen pelin, vaikkei pelimoottoria ole siihen suunniteltu alun perin.

Unreal Engine 4 -pelimoottorin käyttäminen on ilmaista ja julkaistusta pelistä pitää maksaa viisi prosenttia tekijänoikeuskorvauksia, kun ensimmäinen 3000 euroa on tienattu per tuote vuodenneljänneksellä(Unreal Engine 2015).

6.2.4 Libgdx

Libgdx-pelimoottoria vaivaavat yhä samat asiat, miksi me sen alun perin hylkäsimmekin. Siinä ei ole graafista käyttöliittymää ja sillä kehittäminen on paljon hankalampaa kuin muilla mainitsemistani pelimoottoreista jos ei ole laajaa osaamista ohjelmoinnista.

Libgdx on avoimella lähdekoodilla toteutettu pelimoottori ja sen käyttäminen ja sillä julkaiseminen on täysin ilmaista. Libgdx-pelimoottori tukee tällä hetkellä seuraavia alustoja julkaisun suhteen(Bad Logic Games 2015):

- Windows
- Linux
- Max OS X
- Android (2.2+)
- BlackBerry
- iOs
- Java Applet
- Javascript/WebGL

Joten kaikki yleisimmät alustat on tuettu Windows Phone alustaa lukuun ottamatta.

6.3 Kehitysehdotukset

Kuten aiemmin mainitsin, iteroimme lukuisia kertoja ja pelin kehitys venyi hyvinkin pitkälle aikavälille, eikä edistysaskelia tullut saavutettua kokonaisvaltaisesti kovinkaan paljon. Suurin osa ajasta meni kontrollien hiomiseen, koodin päivittämiseen, tekoälyjen paranteluun jne. Proseduurillisten kenttien generoimisen tuomiin kenttäsuunnittelullisten haasteiden selvittämiseen jne.

Pelin vihollisten tekoäly oli hyvin hankala toteuttaa siitä syystä että kentät olivat proseduurillisesti generoituja, monet ideat jotka näyttivät paperilla hyvältä toimivat kehnosti peliin implementoitaessa juuri kyseisestä syystä. Kun asiat eivät edenneet suunnitelmien mukaan ja pelin edistyminen pysyi paikoillaan, joka poisti kehityksen mielekkyyttä ja hidasti sen etenemistä entistä enemmän. Esimerkki tästä on ensimmäisessä kentässä vihollisena toimiva pyörivä vihollinen, jonka oli tarkoitus tippua taivaalta ja laskeutua tason päälle jolloin pelaajan piti hypätä tämän yli. Koska kentät olivat proseduurillisesti generoituja, vihollinen oli harvemmin uhkana pelaajalle, se joko ehti kierimään pois tasolta ennen kuin pelaajalla oli edes mahdollisuutta osua siihen, tai sitten se vaan putosi kahden tason välistä, olematta sen enempää uhaksi.

7 Yhteenveto

Mobiilitasohyppelypelin kontrollit pystyy luomaan eri tavalla ja immersion luomiseen pystyy käyttämään erilaisia menetelmiä. Kontrollien toimivuuden tärkeydestä ei tasohyppelypelissä voi kiistellä, mutta immersio mobiilitasohyppelypelissä pelin menestyksen kannalta ei välttämättä ole kaikkein tärkein aspekti.

Virtuaaliset kontrollit ovat toteutettavissa hyvin, mutta nopeatempoisessa pelissä niiden implementoiminen voi olla haastavaa jos haluaa säilyttää reagoivan ohjattavuuden pelissä. Tämä ei kuitenkaan poissulje sitä, etteikö niitä voisi toteuttaa hyvin nopeatempoisessakin pelissä, kuten esim. Mikey Shorts -pelissä.

Immersiota ilmenee alkeellisella tasolla kaikissa peleissä, joita käytiin lävitse tässä opinnäytetyössä, mutta vaikuttaa siltä, etteivät kehittäjät kiinnitä siihen kovinkaan paljon huomiota. Kuten aikaisemmin maintsin useimmissa läpikäydyissä peleissä sen täysivaltainen poistaminen ei välttämättä heikentäisi pelikokemusta lainkaan.

Pelimme suunnitelmalliset asiat selkenivät kehitysprosessin aikana ja eri elementit pelissä muuttuivat hyvin paljon ja aikaa meni enemmän prototyyppien luomiseen, kuin pelin valmiiksi saamiseen. Jos olisimme jatkaneet vanhalla kaavalla, niin pelistä olisi todennäköisesti tullut huonompi kuin mitä siitä nyt tulisi valmistuessaan lukuisten prototyyppien jälkeen. Jäljelle jäi kuitenkin vielä paljon ratkaisemattomia ongelmia, jotka pitää ratkaista ennen kuin jatkamme pelin kehitystä. Näitä ovat esimerkiksi mielenkiintoisten ja vaihtelevien vihollisten luominen, kun kentät on proseduurillisesti generoituja.

Näitä ongelmia ei ilmennyt tarkastelluissa peleissä samalla tavalla, koska proseduurillisesti generoituja kenttiä oli pelkästään Megatroid-pelissä ja siinäkään kenttiä ei luotu lähestulkoon yhtä satunnaisesti kuin meidän pelissä. Myös se, että hahmoa ei liikutettu automaattisesti horisontaalisessa suunnassa antoi todennäköisesti enemmän vapauksia esimerkiksi vihollisia suunnitellessa.

Immersion suhteen koen onnistuneemme paljon paremmin kuin aikaisemmissa pelissä juurikin vahvan yllätysarvon ansiosta maailmojen vaihtuessa. Myös musiikin monipuolisuus oli meidän pelimme vahvuuksia ja samankaltaisia toteutusmenetelmiä musiikin suhteen ilmeni vain Leo’s Fotrune –pelissä.

En silti voi todeta etteikö tästä koko kehitysprosessista olisi ollut todella paljon hyötyä minun tulevaisuuteni kannalta. Olen oppinut paljon miten asioita ei pidä tehdä ja miten projektinhallinnan pystyy toteuttamaan huonosti etäympäristössä mahdollisimman monella tavalla. Tulevaisuudessa projekteissa joita lähtee kehittämään, on taustalla paljon enemmän tietoa ja kokemusta siitä, miten välttää paljon erilaisia ongelmia mitä pelinkehityksessä voi ilmetä. Tärkeää on esimerkiksi kehittää prototyyppejä mahdollisimman aikaisessa vaiheessa, jotta saa aikaisin hiottua kaikki perusmekaniikat kuntoon. Tämä helpottaa jatkokehityksen etenemistä ja selviää mahdollisimman aikaisessa vaiheessa, onko peli lainkaan toteutuskelpoinen.

Lähteet

1337 & Senri LLC. 2014. Leo’s Fortune info page. Google Play. https://play.google.com/store/apps/details?id=com.leosfortune.

2.9.2015.
Amogh Timsina. 2015. Top Android Platformer games.
http://appslova.com/top-android-platformer-games/.
16.11.2015.
Bad Logic Games. 2015. Features.
https://libgdx.badlogicgames.com/features.html.
16.11.2015.
Bogdan, P. 2014. Games vs. Hardware. The History of PC video games: The 80's.
Frogmind. 2015. Badland Website.
http://badlandgame.com/.
16.11.2015.
Gamesradar_US. 2010. Gamings Most Important Evolutions.
http://www.gamesradar.com/gamings-most-important-evolutions/?page=3.
16.11.2015.
Jamie Madigan. 2010. Analysis The Psychology of Immersion in Video Games.
http://www.gamasutra.com/view/news/120720/Analysis_The_Psychology_of_Immersion_in_Video_Games.php.
16.11.2015.
Joe Hindy. 2014. 12 best Android platform games.
http://www.androidauthority.com/best-android-platform-games-528118/.
16.11.2015.
Joseph Leray. 2013. Badland review.
http://toucharcade.com/2013/04/12/badland-review/.
16.11.2015.
Mariowiki. 2015. Super Mario Bros.
http://www.mariowiki.com/Super_Mario_Bros.
2.9.2015.
Mark Brown. 2012. Mikey Shorts review.
http://www.pocketgamer.co.uk/r/iPad/Mikey+Shorts/review.asp?c=44507.
16.11.2015.
Mediocre. 2014. Granny Smith info page. Google Play.
https://play.google.com/store/apps/details?id=com.mediocre.grannysmith.
2.9.2015.
Noodle Cake Studios. 2015. Mikey Shorts info page. Google Play.
https://play.google.com/store/apps/details?id=com.noodlecake.mikeyshorts&hl=fi.
2.9.2015.
Odoerfer, M. 2014. Jungle Hunt Was A Terrible Waste Of Quarters.
http://retrovolve.com/jungle-hunt-was-a-terrible-waste-of-quarters/.
16.11.2015.
Rignall, J. 1990. Ducktales Mean Machines Review
http://www.meanmachinesmag.co.uk/review/101/duck-tales.php.
16.11.2015.
Robi Gangulay. 2015. The 5 Biggest Mistakes in Mobile App Marketing.
https://blog.kissmetrics.com/mistakes-in-app-marketing/.

16.11.2015.
Ryan Whitwam. 2014. Leo's Fortune Review: An Action Platformer To Treasure. http://www.androidpolice.com/2014/07/09/leos-fortune-review-an-action-platformer-to-treasure/. 16.11.2015.
SilverTree Media. 2015.Cordy 2 info page. Google Play. https://play.google.com/store/apps/details?id=com.silvertree.cordy2. 16.11.2015.
Stan's Assets. 2015. Android Native Plugin. Unity Asset Store. https://www.assetstore.unity3d.com/en/?gclid=Cj0KEQiA96CyBRDk5qOtp5vz8LkBEiQA6wx8MOV-gCcYr1sKbt2qC_KsaSwsYCdju4kb7oJitUKuNDYcaAg3S8P8HAQ#!/content/10825 16.11.2015.
Triolith Entertainment AB. 2013. Megatroid info page. Google Play. https://play.google.com/store/apps/details?id=com.Triolith.MEGATROID&hl=fi. 2.9.2015.
Unity Technologies. 2015c. Buy unity pro. https://store.unity3d.com/?gclid=CjwKEAiA1JuyBRCo-gJLz4J71kj0SJADsd6QRxymW1esPVYsh9tdwUe-incOi7Av598oAgXEo62e4F_BoCfJTw_wcB 16.11.2015.
Unity Technologies. 2015b. End User License Agreement. https://unity3d.com/legal/eula 16.11.2015.
Unity Technologies. 2015a. Unity3D page index. http://unity3d.com/unity. 16.11.2015.
Unreal Engine. 2015. Index. https://www.unrealengine.com/what-is-unreal-engine-4. 16.11.2015.
YoYo Games. 2015b. Buy studio pro. https://www.yoyogames.com/buy/studio_pro. 16.11.2015.
YoYo Games. 2015c. Buy. https://www.yoyogames.com/buy/. 16.11.2015.
YoYo Games. 2015a. Index. http://www.yoyogames.com/. 16.11.2015.